Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Prototype Development of Risk Mitigation for Software Anti-Ageing System

Thamaratul Izzah Azman[1,*], Noraini Che Pa[1], Rozi Nor Haizan Nor[1], Yusmadi Yah Jusoh[1]

[1] Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Malaysia

ARTICLE INFO

Article history:
Received 15 January 2023
Received in revised form 23 June 2023
Accepted 30 June 2023
Available online 16 July 2023

Keywords:
Risk mitigation; software anti-ageing; software ageing; software engineering; software maintenance

ABSTRACT

Constance computer program changes during software maintenance cause program structure and its execution to degrade which in the long run reduce the quality of the program, driving to the rise of software ageing. Change analysis is essential to assess and oversee the effect of changes to handle ageing issue. However, the evaluation of risks is still vague. There are also insufficient tools for change analysis amid software maintenance for ensuring anti-ageing. This motivates the study to propose risk mitigation process as a method to evaluate the risks from software change and construct a prototype as a platform to aid change analysis phase during software maintenance. The prototype aims to foresee and minimize chances of risks from software changes that lead to software ageing. This paper discusses on the development of a prototype named Risk Mitigation for Software Anti-Ageing System that is designed and developed based on software development lifecycle methodology to establish successful development and implementation of the system. The discoveries from this study offer assistance for software maintainers to conduct risk assessment during change analysis in software maintenance via the digital risk mitigation process to ensure software anti-ageing.

1. Introduction

Software changes are essential to rectify faults and upgrade the software with new attributes and functions. As software advances, new changes for existing software may come along with risks [1,2]. The increasing software size and its complexity magnifies software errors to the point where effective risk assessment is critical [3,4]. Failure to oversee risks will influence the success of maintenance causing software failures, decrease execution performance and diminishes the benefits from the computer program leading to early software retirement (Salmeron and Lopez). This situation leads to software ageing which is characterized as degradation of computer program execution, performance and quality influencing the capacity of the software to function and deliver its services normally [5,6]. Parnas [7] highlighted that software ages as a result of software product's proprietor

* *Corresponding author.*
E-mail address: eizahazman@gmail.com

https://doi.org/10.37934/araset.31.2.220233

failure to alter it to its changing needs or from the outcome of changes made. Impacts from software ageing may raises challenges to manage with business operation as it moderates down computer program response time to transact commands resulting to postponed works and reduces quality of services [8-10].

In particular, adaptable and flexible software features helps ensuring the software to remain anti-ageing albeit future software changes [11]. Software anti-ageing portrays computer program that keeps up its quality and is adaptable to any future modification permitting it to stay significant in its environment [12]. Software ageing from software engineering perspective concerns with assessing and overseeing consequences from software change that contributes to software ageing (Russo). It is highly crucial for maintainers to examine any risks or threats from software change during change analysis phase in software maintenance [13]. In spite of its significance, the assessment of risks is still vague during change analysis [14]. There is also scarcity of tool or instrument for aiding maintainers during change analysis to ensure software to remain anti-ageing [12,15]. Hence, this drives the research to propose risk mitigation process as a strategy to assess risks during change analysis and create a prototype as a new tool that aids maintainers for managing and reducing possible threats to software ageing during software maintenance for attaining software anti-ageing. The paper structure begins with reviewing related literature, method for the study, discussion on the findings and lastly, concluded with a summary and further work.

2. Related Literature

Regular changes amid software maintenance bring about risks to deteriorate software structure and quality causing software ageing [16,17]. Managing and determining consequences from software change are often performed during change analysis in software maintenance [18].

Technology readiness is significant for any change management in an organization to further optimize performance [19]. Therefore, conducting change analysis to ensure this is imperative for a successful software maintenance, however, there are few challenges the way changes should be handled and administered [20]. One problem arisen is on finding correct and suitable approach for maintainers to analyze the risks impact from software changes [13]. The majority of current change analysis methods focus on determining the impact of changes in source code [20]. Organizations however are more inquisitive on viable management of risks that cover more than fair bugs within the code [21]. A study in [2] suggested risk mitigation through architecture evaluation to alleviate and reduce the risk of software changes. The strategy guides for decision-making process on future software changes through assessing the program design. However, the scope of moderating the risks within the study is equivocal because it centered on subjective in-depth understanding of the software architecture, which is poorly scalable. This issue of lacking quantifiable scale to measure the effect of risks motivates the study to develop a scalable risk mitigation process for change analysis.

The objective of risk mitigation is to decide strategic method for lessening the effect from risks in software [22]. It comprises of four major processes such as identification of risk, risk decision, treating the risks and risk monitoring [23]. Risk identification is the first step in risk mitigation where risk is recognized through assembling information and data, evaluating important risks area and implementing analytical tools to diagnose possible risks [24]. Risk decision is then performed. This phase is remarkably crucial where risks are measure according to the degree of its likelihood and impact and prioritization of distinguished risks into their consecutive levels [23,24]. Decision makers may gauge and estimate risk chances and magnitude by utilizing definitive benchmark established [23].

Next, risks are treated through risk solution and action according to its magnitude. Risk treatment comprises of several treatment options such as avoidance, reduction, transfer, acceptance and eliminate [24]. Lastly, treated risks are then monitored to help checking and keeping track of former risks. This phase is vital to report, review and oversee action taken on risks [24]. Risks are monitored through regular risk profile review, risk reassessment and revising hazard profile with executed action [25]. It also helps to observe whether the risk magnitude is successfully reduced to a satisfactory level and scrutinizes the effectiveness of mitigation action taken [26].

Several systems have been developed and proposed by existing studies where it acts as a tool or instrument to reduce the effort required to perform risk mitigation and operates as a knowledgebase for storing and retrieving risk information. Sun *et al.,* [28] proposed a web-based decision support system to aid risk decision-making in predicting risk chances level. It was constructed using decision support framework that incorporates sub-systems such as data administration, a defined user interface, selection, management and model evaluation. The system performs calculations to generate risk levels for risk decision-making in three risk grades: low risk (safe), medium risk, and high risk. Orlando *et. al.,* built a web-based risk assessment and clinical decision support programme to aid in the adoption of risk-based rules that help reduce costs and improve care quality. It aids in the reduction of risk assessment gaps such as risk overestimation and underestimation.

Researcher in [23] established a risk mitigation mechanism to aid in the reduction of IT governance risks. It involves mapping of knowledge for knowledge base and includes several software agents for risk estimation in decision-making. It also generates suggestions for risk monitoring and facilitates expert collaboration in decision-making. Meanwhile, a web-based virtual risk management system built by Sun *et al.,* [28] incorporates risk reduction processes in accordance with various virtual enterprises. There is multi-agent utilized in the system. Via knowledge and rules of expert, it produces explicit risk mitigation process for aiding decision-making. [29] created a spatial decision support framework for software development project risk mitigation. The system incorporates diverse modules such as input module, risk assessment module, decision analysis module and visualization module. Its goal is to find a feasible and user-friendly way to perform decision-making and risk mitigation to ensure project's cost and time requirements are reduced.

From the literature, the lack of adequate mechanisms or tools in current practice to assist maintainers with change analysis during software maintenance to achieve anti-ageing arises a significant challenge [12,15]. Notably, maintainers often oversee and handle change analysis report using a straightforward web interface database that only allows for browsing and searching for risk reports [15]. As a result, there is concern about a lack of competent instruments for change analysis. Existing risk mitigation process for change analysis proposed by past researchers appears to be ambiguous as it is not scalable and quantifiable as it centered on qualitative assessment for risk mitigation [2]. Existing systems are also insufficient to provide a complete risk mitigation process for change analysis in software maintenance as few of them are only concentrates on decision-making operation. Some of the systems disregard the process of risk identification, excluded recommendations for risk action for risk treatment and eliminate plan for risk monitoring. On that account, the existing systems are still lacking complete risk mitigation process.

Therefore, the study captured the requirements from the criteria in the existing systems to facilitate the prototype development and outline the functions to be integrated in the new system. The purpose of the study is to provide state-of-the-art tool to examine, handle and monitor risks associated with software changes using risk mitigation process to keep the software safe from negative effect that would jeopardise its quality and performance.

3. Methodology

This section discusses on the method adopted and step taken for the study. For prototyping purposes, software development lifecycle (SDLC) methodology is applied such as requirement analysis, software design, software implementation, software testing and software maintenance [30, 31]. It offers structural sequence of steps and tasks to be conducted for each phase of the development. Figure 1 shows five phases involved in software development lifecycle (SDLC) methodology.

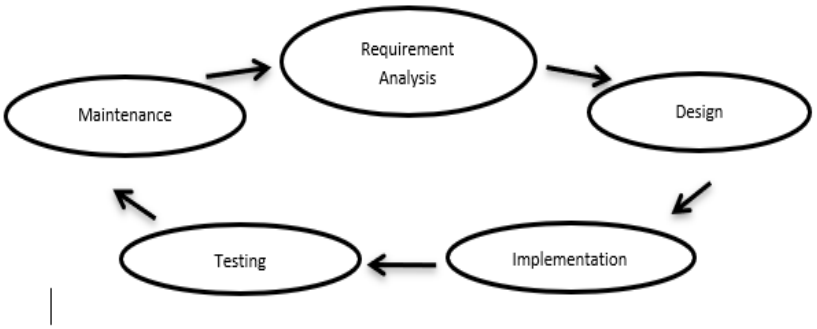

Fig. 1. Software Development Lifecycle (SDLC)

For the first phase of prototype development, the study carries out software requirement analysis through literature review. Requirement analysis is crucial in ensuring successful software project development as it concerns with determining the needs for the project [32]. Requirements are function description of software should incorporate in its computer program that involve with system response to inputs and its behaviour in a particular circumstance [33].

Software design is related with the development of prototype architectural design through use case diagram, class diagram and software system architecture. These diagrams are designed to overview the prototype's functions, behaviour and its environment. To portray processes flow in the prototype, graphical user interface (GUI) map is built. Once the prototype coding is completed according to the prototype design, it is then run in local host for implementation phase. Its interfaces are presented to demonstrate the processes embedded. Lastly, the prototype undergoes system testing to ensure the system operates as it is intended to be. Testing process starts from unit testing until the overall system performance where three experts were engaged in the process that includes two software developers and one software project manager. The findings for each phase in software development lifecycle (SDLC) will be discussed in the next section.

4. Findings and Discussion

4.1 Requirement Analysis

To achieve the objective of the study, literature review was conducted to gather and capture the requirements needed for the development of system. The study had identified and assembled the requirements for prototype development from existing risk mitigation systems in the literature. Table 1 depicts the comparison of the criteria in each risk mitigation systems. Based on Table 1, the study discovered that risk mitigation processes in most of the systems are incomplete. The systems built in [23, 27, 29, 34] excluded risk identification feature in their systems. This is possibly because manual process of risk identification through meetings or brainstorming may be adopted to determine the probable risks. Moreover, the system developed by Roya *et. al.,* disregard risk treatment feature to specify risk best suggestion and action plan for the identified risk. Hence, because of this comparison,

the processes in the existing systems are still incomplete in depicting the entire risk mitigation process.

Table 1
Criteria comparison between existing systems

Criteria/ Existing Systems	[27]	[34]	[23]	[29]	[28]
Risk identification					✓
Manage risks information		✓	✓		✓
Risk decision	✓	✓	✓	✓	✓
Risk database	✓	✓	✓	✓	✓
Risk best suggestion and action plan		✓	✓		✓
Report generation	✓	✓	✓	✓	✓
Comment			✓		
Authentication			✓	✓	
Monitor risks			✓		

There are four processes in risk mitigation that includes risk identification, risk decision, risk treatment and risk monitoring [23,35]. Thus, a set of the requirements for the prototype built in this study should include those four risk mitigation processes. Table 2 describes several prototype requirements in which the functions in the prototype are developed according the requirements listed below.

Table 2
Prototype requirements

Requirements	Description	Source
Identify risk	Providing function for risk identification	[28]
Risk decision	Providing function to perform risk decision	[23, 27, 28, 34]
Risk best practice suggestion	Providing function that suggests best practice suggestion for risk treatment	[23, 29, 34]
Monitor risk	Providing function for monitoring risks	[23]
Risk report	Providing function that generate risk report	[23, 27, 29, 34]
Manage information	Providing function for managing information about the risk (e.g risk name, risk description, risk suggestion)	[23, 28, 34]

4.2 System Design

From the requirements described, the study constructed the prototype design to illustrate its behaviour, functions, and environment. It includes constructing the use case diagram, class diagram and system architecture. Figure 2 depicts the prototype's use case diagram. The use case diagram is used to visualise the prototype's functions and the key users involved in executing those functions. Two main users involved in this system include maintainer and staff. Both maintainer and staff may register and authenticate themselves before being directed into the system. Maintainer may perform complete features of risk mitigation process such as identifying risk for risk identification, identify risk impact and probability to conduct decision-making process for risk decision, retrieve plan action for risk treatment, monitor risk and view risk report as well as managing risk information. Meanwhile, staff can only access a limited number of functions in the system, such as displaying risk data and generating risk reports.

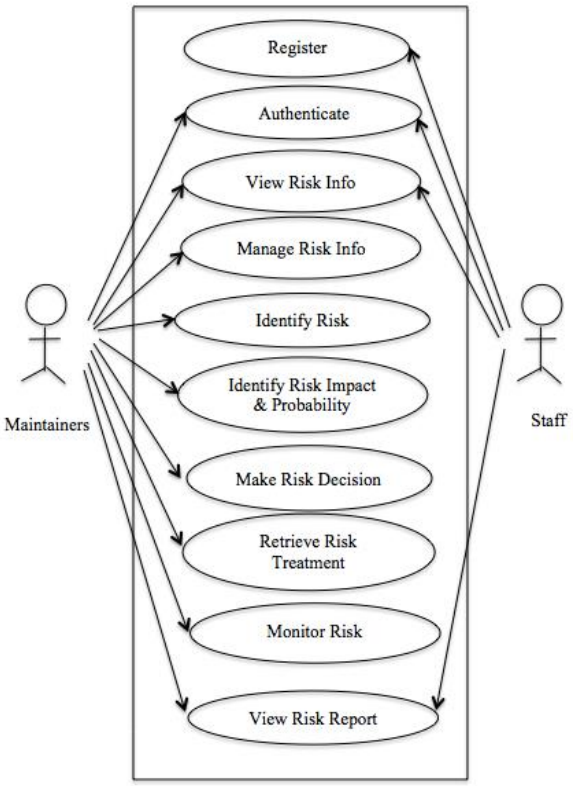

Fig. 2. Use case diagram of Risk Mitigation for Software Anti-Ageing system

A class diagram is constructed to outline the relationship between classes in the prototype. UML class diagram helps describing relationships and attributes between variables for system design [36]. Each class in the class diagram is constituted of three tiers: class name (upper tier), class attributes (middle tier) and class methods (bottom tier) [37]. Figure 3 portrays the prototype's class diagram. The class diagram includes six classes: maintainers, staff, risk information, risk mitigation, risk monitoring, and risk report. Environment in the system is illustrated through the class diagram where a maintainer may conduct risk mitigation and risk monitoring as well as manages risk info and risk report and on the other hand, staff may view only risk info and risk report.

System architecture is intended to depict the overall structure and behaviour of a system. It incorporates system components such as hardware and software components, network components and database component with its environment. Figure 4 depicts the prototype's system architecture. It constitutes three layers of environment such as client layer, service layer and database layer. Users communicate with the device via hardware components such as a monitor and software components such as a browser through a web-server in the client layer. Risk Mitigation for Software Anti-Aging System is a service layer that includes its own functions through an application server. Finally, the database layer includes database components that store the system's data and information.

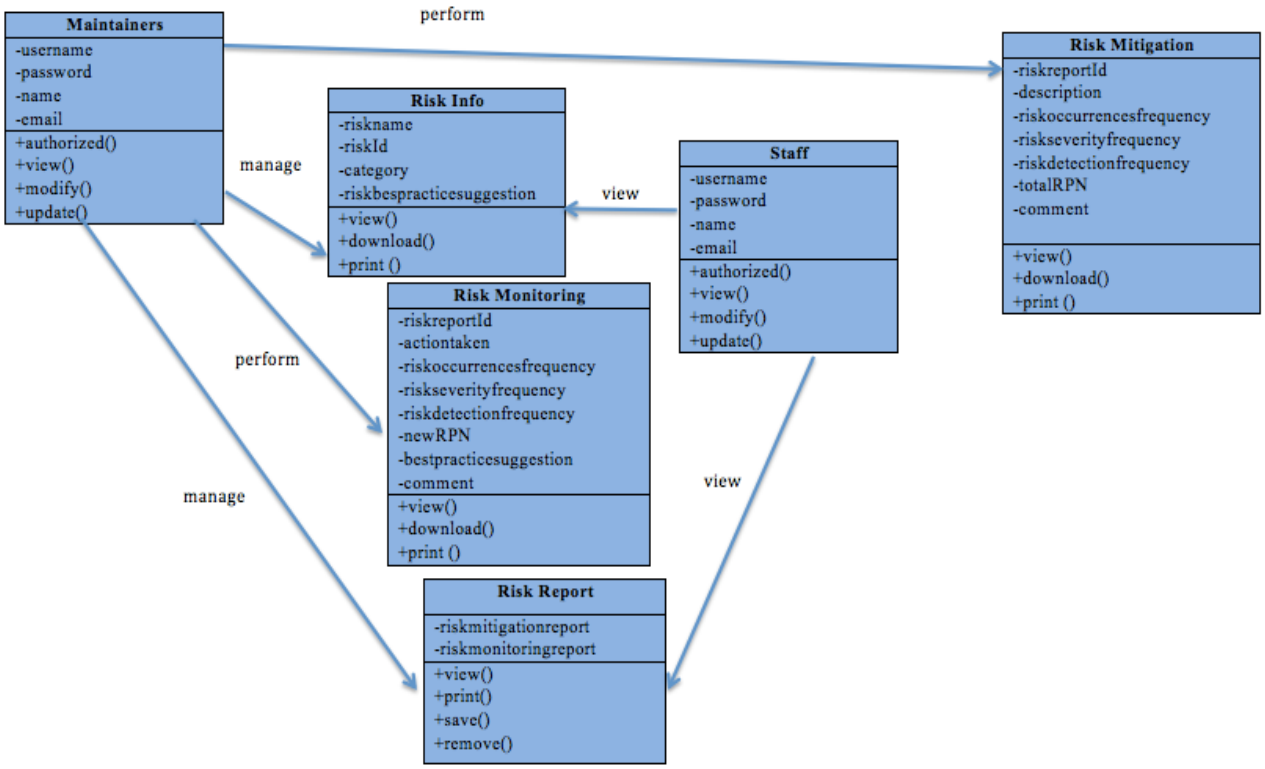

Fig. 3. Class diagram of Risk Mitigation for Software Anti-Ageing system

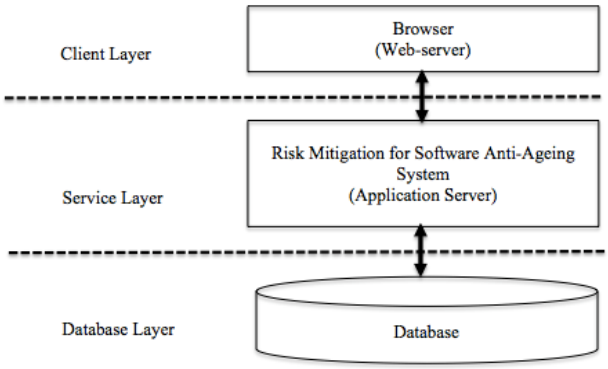

Fig. 4. Risk Mitigation for Software Anti-Ageing system architecture

4.3 Prototype Implementation

The prototype undergoes several steps of designing its UI, coding and run it locally on a XAMPP server. The system is coded on Eclipse IDE using CSS, PHP and Javascript scripting and runs on XAMPP local host server. After the completion of prototype coding, the prototype is implemented to realize its architectural design. The functions and features in the prototype were based on the requirements captured and specified in the previous section. A prototype named Risk Mitigation for Software Anti-Ageing system is then successfully constructed. The study illustrates the system interface using a graphical user interface (GUI) map to provide an overview of the system's flow as in Figure 5.

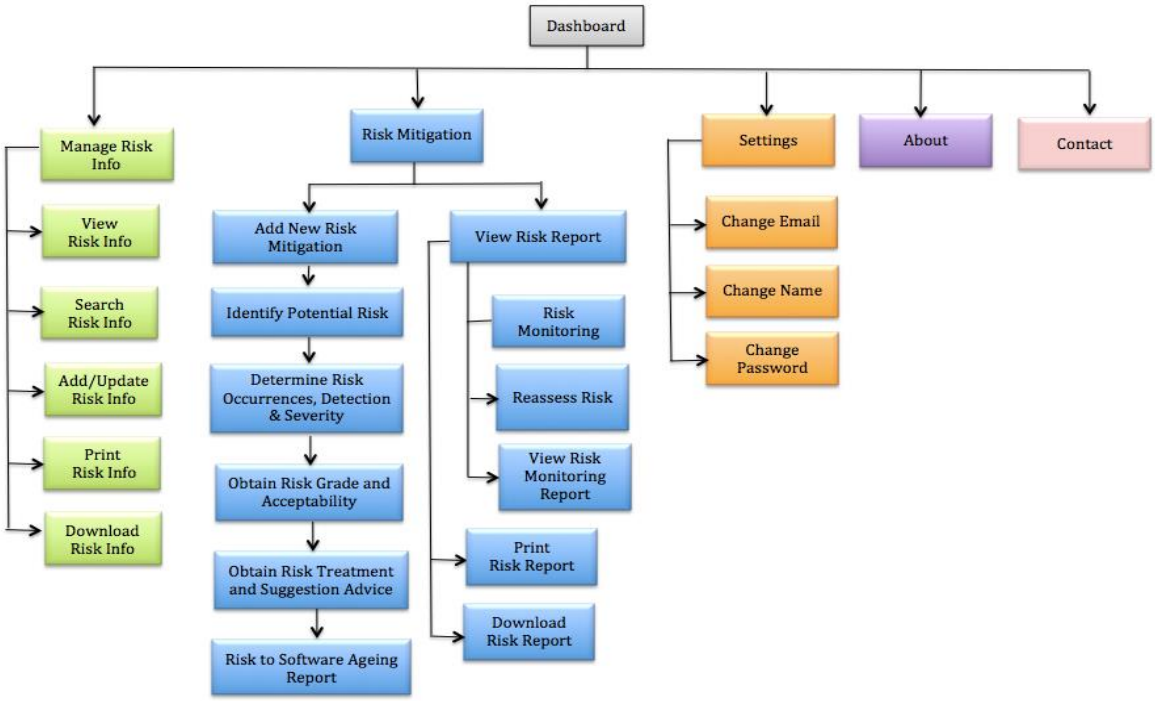

Fig. 5. Graphical User Interface (GUI) map

The GUI map is intended to depict the navigations of each function and interaction within the system interface starting from the dashboard. There are five main tabs in the dashboard (homepage) that includes 'Manage Risk Info', 'Risk Mitigation', 'Settings', 'About' and 'Contact'. Features such as viewing risk information, updating, printing, and downloading risk information can be done in 'Manage Risk Info' tab. 'In Risk Mitigation' tab risk mitigation process is performed such as determining potential risk for risk identification, estimating risk occurrences, detection and severity for making risk decision, acquiring risk grade and acceptability, getting suggestion action plan for risk treatment and lastly generating risk mitigation report. Risk mitigation report also can be viewed, downloaded, printed under this tab. For risk monitoring, the report shall be monitored through risk reassessment feature under risk mitigation tab, which then generated risk monitoring report. On the other hand, 'Settings' tab allows users to update their username, name, and password. 'About' and 'Contact' tabs are also provided in the system for general information.

Figure 6 depicts the screenshot of system dashboard, which is constructed to be the main homepage of the system. The vertical navigation menu in the system comprises of 'Dashboard' tab, 'Manage Risk Info' tab, 'Risk Mitigation' tab, 'Settings' tab and 'Log Out' tab meanwhile horizontal navigation menu includes 'About' and 'Contact' tabs.

Fig. 6. Dashboard of the system

'Risk Mitigation' tab is primary feature in the system. It offers maintainer with the functions to determine risks for risk identification, measure risks exposure and its magnitude for risk decision and retrieving risk suggestion advice for treating the risks as well as monitoring the former risks through risk reassessment. Risk mitigation process is initiated once maintainer select risk mitigation form for risk identification, refer Figure 7.

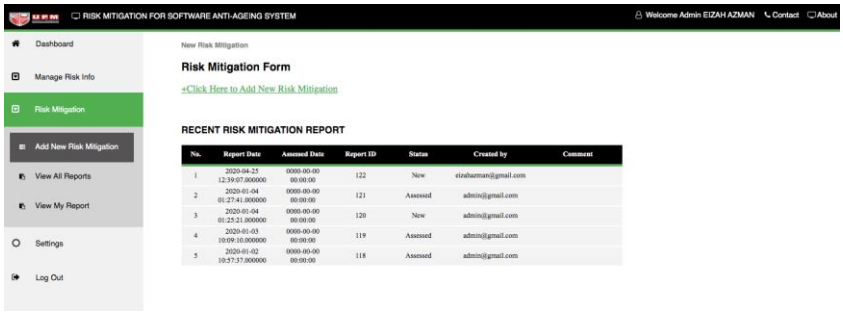

Fig. 7. Risk Mitigation page

The system prepared a set of checklists that consists software change risks that could influence software ageing as shown in Figure 8. The checklist helps the maintainer to capture and differentiate possible risks that they want to evaluate and determine their effect based on their knowledge and judgment. The checklist includes a list of risk categories, risk names, and risk descriptions to aid in weighing their arguments.

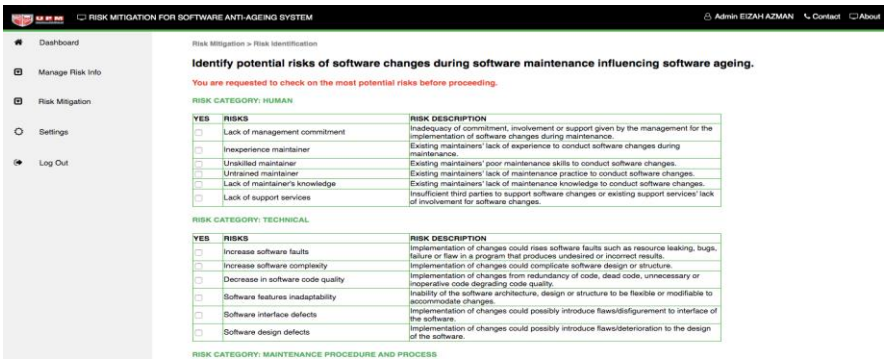

Fig. 8. Risk Identification page

The maintainer will be guided to the risk decision page once the risks have been identified as in Figure 9. The frequency of occurrence, identification, and severity of identified risks will be quantified using an array of values to quantify its probability and effect. Each defined risk will be rated according to its "occurrence of failure," which indicates the likelihood that the risks will occur as a consequence of a particular cause, "severity," which indicates the magnitude of the potential risks' impact on the process of performing changes when they occur, and "detection," which indicates the likelihood that a potential failure from the risks will be detected by using a 5-point scale ranging from '1' denoting low frequency to '5' denoting high frequency. The failure mode estimation analysis technique is used to make decisions by multiplying risk occurrence, severity and detection frequency. As a result, a Risk Priority Number (RPN) is produced, which aids in assigning the effect of the identified risks to their level of exposure and magnitude.

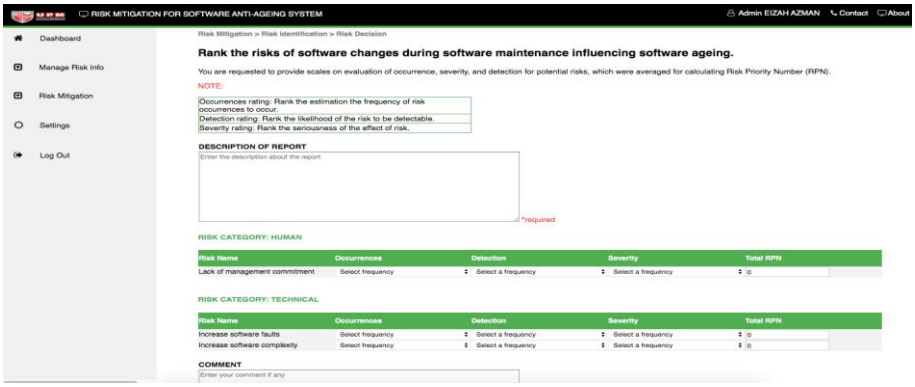

Fig. 9. Risk Decision page

The risk grade and acceptability are then retrieved, and risk treatment advice is provided based on the Risk Priority Number (RPN) value. The value of the Risk Priority Number (RPN) is defined using a score benchmark adopted from [38]. Based on the RPN value, a risk grade and acceptability are allocated with different colours to reflect the degree of severity and acceptability. Figure 10 depicts a risk mitigation report that includes the identified risks, RPN value, risk grade and acceptability, and risk treatment advice. The risk treatment advice is created automatically from the knowledgebase (the system's database) based on the risk grade and acceptability value. These suggestions came from interviews with a number of experts in the field. Experts were asked to define appropriate recommendation advice for each risk based on five RPN value categories. If risk treatment requires future adjustments, the knowledgebase's risk treatment advice can be updated through the 'Manage Risk' tab.

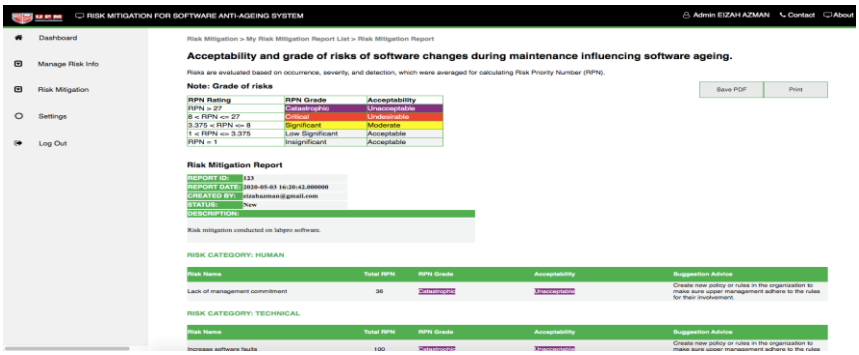

Fig. 10. Risk Report page

'Risk Mitigation' tab also incorporates feature to reassess treated risks for risk monitoring purposes.

Risk reassessment involves revisiting previously defined and mitigated risks and applying a similar strategy of multiplying risk occurrence, detection, and severity frequencies to determine a new risk grade and acceptability level. Figure 11 illustrate the risk reassessment results for the risk monitoring report, which include the previously identified risk, the old and new RPN value, new risk grade and acceptability, as well as new risk treatment recommendations based on the new RPN value.

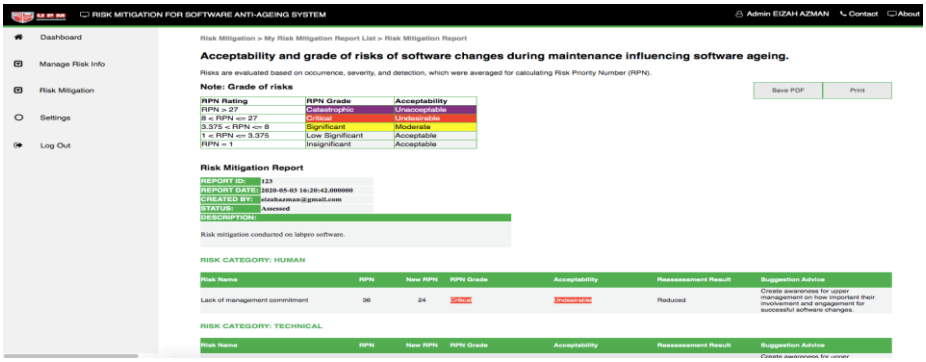

Fig. 11. Risk Monitoring report page

4.4 Prototype Testing

The objective of prototype testing is to ensure that the system and its processes operate smoothly and completely. For testing purposes, three testers were involved in conducting unit testing to inspect the codes for each of the functions in the prototype and perform the overall system interface testing for evaluating the whole system performance and its compliance with specified requirements. The testers discovered no errors that affect the system's operation based on the prototype testing results, and the overall system interface testing was found to operate smoothly in accordance with its navigations without any response pause, glitch, or loading failure. Figure 12 depicts one of the prototype testing results.

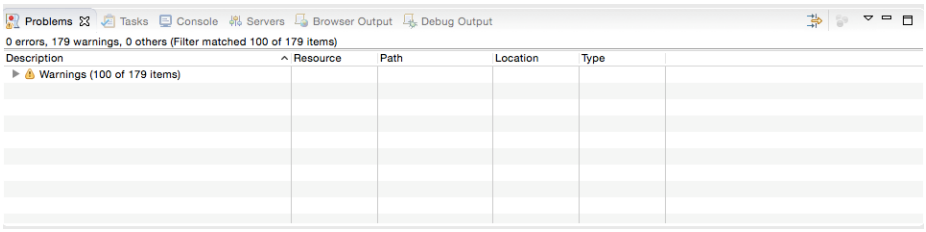

Fig. 12. Unit testing report

Overall, using the software development lifecycle approach, a prototype of Risk Mitigation for Software Anti-Aging System was successfully developed. The system developed in this study addresses the issue raised by [12,15] regarding the lack of tools or mechanisms to aid in change analysis for ensuring software anti-ageing. The system serves as a platform or tool for software maintainers to execute risk mitigation process during change analysis in software maintenance to ensure software in an anti-aging state after performing software change. It incorporates a complete and in sequence risk mitigation process embedded in the system to cater the insufficient risk mitigation processes in the existing risk mitigation systems. The system also offers a scalable and quantifiable measurement of risk compares to risk mitigation suggested by Knodel and Matthias [2] which concerns on qualitative evaluation that is quite ambiguous. It also includes numerical information on the risks that presents clear risk interpretation to assure the accuracy of its information. Risk mitigation process that is scalable reduces bias in the interpretation of risk magnitude as it offers a more rigid and straightforward information on risk decision compared to qualitative approach. Quantifiable approach provided in the system also allows specific and fixed data results for risk report, providing maintainers with a more accurate value of the risks magnitude. This helps restraining the chances of ambiguous and misinterpretation of risk exposure and its magnitude. In general, the system built in this study is useful and valuable for maintainers in

providing a digitalized risk mitigation process to handle risk of software change during change analysis in software maintenance for achieving software anti-ageing.

5. Conclusion

In conclusion, the issue of incorporating risk assessment for change analysis during software maintenance is resolved by proposing a risk mitigation process for managing and mitigating risks. Concerns about the insufficient of tool for support of maintainers during change analysis to achieve software anti-ageing are also addressed by developing a prototype of Risk Mitigation for Software Anti-Ageing system. Concerns regarding the lack of a tool to assist maintainers during change analysis in order to achieve software anti-ageing are also addressed through the development of the prototype. The prototype was designed and developed to incorporate four major processes in risk mitigation and requirements reviewed from existing risk mitigation systems. It performs several functions, including determining risks for risk identification, quantifying risk likelihood and impact for making risk decision, retrieving suggestion advice for risk treatment, reassessing risks for risk monitoring, generating risk reports and managing risk information. Generally, the prototype's implementation leads to a new process and tool for change analysis during software maintenance as a new way to combat software ageing. It also contributes to promote a convenient risk mitigation tool through scalable and quantifiable approach for assisting maintainers in handling and tackling the risks of software change for ensuring software state as anti-ageing. In the future, the study shall conduct a qualitative study with software practitioners to validate the prototype's usability and usefulness in practice, as well as verifying the quality of programmes built in the system.

Acknowledgement

This research work is funded and supported by Malaysian Education Ministry, granted under Putra Graduate Initiative Grant, Universiti Putra Malaysia (GP-IPS/2018/9644600).

References

[1] Cotroneo, Domenico, Antonio Ken Iannillo, Roberto Natella, and Roberto Pietrantuono. "A comprehensive study on software aging across android versions and vendors." *Empirical Software Engineering* 25 (2020): 3357-3395. https://doi.org/10.1007/s10664-020-09838-3

[2] Knodel, Jens, and Matthias Naab. "Mitigating the Risk of Software Change in Practice." Interpretation 110 (2014): 01.

[3] Boranbayev, A. S., S. N. Boranbayev, Assel M. Nurusheva, K. B. Yersakhanov, and Erzhan Nurakhanovich Seitkulov. "Development of web application for detection and mitigation of risks of information and automated systems." Eurasian Journal of Mathematical and Computer Applications 7, no. 1 (2019): 4-22. https://doi.org/10.32523/2306-6172-2019-7-1-4-22.

[4] Xiang, Jianwen, Caisheng Weng, Dongdong Zhao, Jing Tian, Shengwu Xiong, Lin Li, and Artur Andrzejakb. "A New Software Rejuvenation Model for Android." 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), October 2018. https://doi.org/10.1109/issrew.2018.00021.

[5] Yahaya, Jamaiah H., Aziz Deraman, and Zuriani Hayati Abdullah. "Evergreen software preservation: The anti-ageing model." In *Proceedings of the International Conference on Internet of things and Cloud Computing*, pp. 1-6. 2016. https://doi.org/10.1145/2896387.2896436

[6] de Melo, Matheus D'Eça Torquato, Jean Araujo, I. M. Umesh, and Paulo Romero Martins Maciel. "Sware: an approach to support software aging and rejuvenation experiments." *Journal on Advances in Theoretical and Applied Informatics* 3, no. 1 (2017): 31-38. https://doi.org/10.26729/jadi.v3i1.2441

[7] Parnas, David Lorge. "Software aging." In *Proceedings of 16th International Conference on Software Engineering*, pp. 279-287. IEEE, 1994.

[8] Oliveira, Felipe, Jean Araujo, Rubens Matos, Luan Lins, André Rodrigues, and Paulo Maciel. "Experimental evaluation of software aging effects in a container-based virtualization platform." In *2020 IEEE International*

Conference on Systems, Man, and Cybernetics (SMC), pp. 414-419. IEEE, 2020. https://doi.org/10.1109/SMC42975.2020.9283358

[9] P., Shruthi, and Nagaraj G. Cholli. "An Analysis of Software Aging in Cloud Environment." International Journal of Electrical and Computer Engineering (IJECE) 10, no. 6 (December 1, 2020): 5985. https://doi.org/10.11591/ijece.v10i6.pp5985-5991.

[10] Chen, Pengfei, Yong Qi, Xinyi Li, Di Hou, and Michael Lyu. "ARF-Predictor: Effective Prediction of Aging-Related Failure Using Entropy." IEEE Transactions on Dependable and Secure Computing, 2016, 1–1. https://doi.org/10.1109/tdsc.2016.2604381.

[11] Yahaya, Jamaiah, and Aziz Deraman. "Towards the anti-ageing model for application software." In *Proceedings of the World Congress on Engineering*, vol. 2. 2012.

[12] Abdullah, Zuriani Hayati, Jamaiah Yahaya, Siti Rohana Ahmad Ibrahim, Sazrol Fadzli, and Aziz Deraman. "The implementation of software anti-ageing model towards green and sustainable products." *International Journal of Advanced Computer Science and Applications* 10, no. 5 (2019). https://doi.org/10.14569/IJACSA.2019.0100507

[13] Rahman, Marfizah A., Rozilawati Razali, and Fatin Filzahti Ismail. "Risk factors for software requirements change implementation." *International Journal of Advanced Computer Science and Applications* 10, no. 3 (2019). https://doi.org/10.14569/IJACSA.2019.0100316

[14] Rahman, Marfizah Abdul, Rozilawati Razali, and Dalbir Singh. "A Risk Model of Requirements Change Impact Analysis." *J. Softw.* 9, no. 1 (2014): 76-81. https://doi.org/10.4304/jsw.9.1.76-81

[15] Borg, Markus, Krzysztof Wnuk, Björn Regnell, and Per Runeson. "Supporting change impact analysis using a recommendation system: An industrial case study in a safety-critical context." *IEEE Transactions on Software Engineering* 43, no. 7 (2016): 675-700. https://doi.org/10.1109/TSE.2016.2620458

[16] Mahmud, Hoger. "A Simple Software Rejuvenation Framework Based on Model Driven Development." *UHD Journal of Science and Technology* 1, no. 2 (2017): 37-45. https://doi.org/10.21928/uhdjst.v1n2y2017.pp37-45

[17] Catolino, Gemma, Fabio Palomba, Andrea De Lucia, Filomena Ferrucci, and Andy Zaidman. "Enhancing change prediction models using developer-related factors." *Journal of Systems and software* 143 (2018): 14-28. https://doi.org/10.1016/j.jss.2018.05.003

[18] Wang, Yibin, Maksym Petrenko, and Václav Rajlich. "Evaluating Heuristics for Iterative Impact Analysis." *arXiv preprint arXiv:1907.08730* (2019).

[19] Shwedeh, Fanar, Norsiah Hami, Siti Zakiah Abu Bakar, Fadhilah Mat Yamin, and Azyyati Anuar. "The Relationship between Technology Readiness and Smart City Performance in Dubai." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 29, no. 1 (2022): 1-12. https://doi.org/10.37934/araset.29.1.112

[19] Isong, Bassey, and Obeten Ekabua. "Towards Improving Object-Oriented Software Maintenance during Change Impact Analysis." In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, p. 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2013.

[20] Lehnert, Steffen. "A review of software change impact analysis." (2011). https://doi.org/10.1145/2024445.2024454

[21] Shihab, Emad, Ahmed E. Hassan, Bram Adams, and Zhen Ming Jiang. "An industrial study on the risk of software changes." In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, pp. 1-11. 2012. https://doi.org/10.1145/2393596.2393670

[22] Shahzad, Basit, Yousef Al-Ohali, and Abdullah Azween. "Trivial model for mitigation of risks in software development life cycle." *International Journal of the Physical Sciences* 6, no. 8 (2011): 2072-2082.

[23] Jnr, Bokolo Anthony, Noraini Che Pa, Rozi Nor Haizan Nor, and Y. J. Josoh. "Knowledge mapping and multi-software agents-based system for risk mitigation in IT organizations." *Journal of Software Engineering and Intelligent Systems* 1, no. 1 (2016): 61-80. https://doi.org/10.15282/ijsecs.3.2017.1.0023

[24] Aloini, Davide, Riccardo Dulmin, and Valeria Mininno. "Risk assessment in ERP projects." *information systems* 37, no. 3 (2012): 183-199. https://doi.org/10.1016/j.is.2011.10.001

[25] Avdoshin, Sergey M., and Elena Y. Pesotskaya. "Software risk management." In *2011 7th Central and Eastern European Software Engineering Conference (CEE-SECR)*, pp. 1-6. IEEE, 2011. https://doi.org/10.1109/CEE-SECR.2011.6188471

[26] Firdose, Salma, and L. Manjunath Rao. "3LRM-3 Layer Risk Mitigation Modelling of ICT Software Development Projects." *International Journal of Electrical & Computer Engineering (2088-8708)* 6, no. 1 (2016). https://doi.org/10.11591/ijece.v6i1.9026

[27] Mukhlash, Imam, Ratna Maulidiyah, and Budi Setiyono. "Web-based decision support system to predict risk level of long term rice production." In *Journal of Physics: Conference Series*, vol. 890, no. 1, p. 012143. IOP Publishing, 2017. https://doi.org/10.1088/1742-6596/890/1/012143

[28] Sun, Xianli, Min Huang, and Xingwei Wang. "Web and multi-agent based virtual enterprise risk management system." In *2011 Chinese Control and Decision Conference (CCDC)*, pp. 902-906. IEEE, 2011. https://doi.org/10.1109/CCDC.2011.5968311

[29] Olyazadeh, Roya, Zar Chi Aye, and Michel Jaboyedoff. "Development of a prototype for spatial decision support system in risk reduction based on open-source web-based platform." In *Conference Paper*. 2013.

[30] Ruparelia, Nayan B. "Software development lifecycle models." *ACM SIGSOFT Software Engineering Notes* 35, no. 3 (2010): 8-13. https://doi.org/10.1145/1764810.1764814

[31] Ali, Firkhan Ali Hamid, Mohd Khairul Amin Mohd Sukri, Mohd Zalisham Jali, Muhammad Al-Fatih, and Mohd Azhari Mohd Yusof. "Web-Based Reporting Vulnerabilities System for Cyber Security Maintenance." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 29, no. 3 (2023): 198-205. https://doi.org/10.37934/araset.29.3.198205

[32] Pileggi, Salvatore F., Antonio A. Lopez-Lorca, and Ghassan Beydoun. "Ontology in software engineering." In *ACIS 2018-29th Australasian Conference on Information Systems*. 2018. https://doi.org/10.5130/acis2018.bp

[33] Sommerville, Ian. "Software engineering (ed.)." *America: Pearson Education Inc* (2011).

[34] Orlando, Lori A., R. Ryanne Wu, Rachel A. Myers, Adam H. Buchanan, Vincent C. Henrich, Elizabeth R. Hauser, and Geoffrey S. Ginsburg. "Clinical utility of a Web-enabled risk-assessment and clinical decision support program." *Genetics in Medicine* 18, no. 10 (2016): 1020-1028. https://doi.org/10.1038/gim.2015.210

[35] Raj Sinha, Pankaj, Larry E. Whitman, and Don Malzahn. "Methodology to Mitigate Supplier Risk in an Aerospace Supply Chain." Supply Chain Management: An International Journal 9, no. 2 (April 1, 2004): 154–68. https://doi.org/10.1108/13598540410527051.

[36] Noviarini, Diena, Mutia Delina, Ananda Mochammad Rizky, Umi Widyastuti, Osly Usman, and Akhmad Yamani. "Early Warning System for Fire Catcher in Rain Forest of Sumatera Using Thermal Spots." *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences* 103, no. 1 (2023): 30-39. https://doi.org/10.37934/arfmts.103.1.3039

[37] Elsayed, Eman K., and Enas E. El-Sharawy. "Detecting Design Level Anti-patterns; Structure and Semantics in UML Class Diagrams." *J. Comput.* 13, no. 6 (2018): 638-654. https://doi.org/10.17706/jcp.13.6.638-654

[38] Zeng, Sai X., Chun M. Tam, and Vivian WY Tam. "Integrating safety, environmental and quality risks for project management using a FMEA method." *Engineering Economics* 66, no. 1 (2010).