Tailoring Software Development Methodologies for Reliability

Mohammad Ahmadi[1], Babak Bashari Rad[2] and Michael Onuoha Thomas[2]
[1]*School of Computing, Engineering and Mathematics, Western Sydney University, Australia.*
[2]*Asia Pacific University of Technology and Innovation, Malaysia.*
m.ahmadi@westernsydney.edu.au

***Abstract*—In recent times, many organizations have sought ways of improving the quality of software products due to the complexity and continuous change in technological trends. These trends have given rise to more sophisticated software systems, which are required for proper functioning at all times. Most research literature proposes tailoring of standard development methodologies due to their inadequacies and inability to meet up with users' needs and system requirements. Reliability engineering has become an approach towards addressing software systems complexity, and also serve as a guarantee towards quality conformance and assurance of software products. In this research paper, the importance of reliability and tailoring is discussed to lay the foundation for the integration of basic reliability engineering techniques into software development.**

***Index Terms*—Methodologies; Quality Assurance; Reliability Engineering; Software Development.**

I. INTRODUCTION

The world we currently live in is getting complicated day by day, with new issues and trends arising daily due to the complexity and needs industrially and technologically. Software development paradigm for decades continues to evolve with much research both industrially and academically, proposing different dimensions and approach towards mitigating the complexity and issues [1], which arises during service delivery in both software development, deployment and also project management execution. Issues like availability, dependability, usability, and reliability are some fundamental quality principles which according to [2], [3-7] must be possessed by every system for effectiveness and efficiency, due to the ever-changing demands in systems and product functionality. The architectural representation of systems and product have over the past years evolved due to technological changes.

New technological paradigms continue to emerge and evolve with cloud computing, internet of things and big data analytics leading the way. The reliance of humans to these technological paradigms has also increased due to the capabilities of these technologies to easily and readily solve or reduce problems encountered in our daily lives and activities [8-13]. Software integrated into these technological trends needs to function at its maximum capabilities to ensure there is no disruption or interruption of service while in usage.

The principle of tailoring ensures that processes guaranteed in software engineering and information technology are adjusted to meet the needs and objectives of service delivery in a software system. Organizations like PMBOK, PRINCE 2, ISO/IEC and IEEE continues to provide and propose mechanisms, principles, and techniques for integration into normal development, production, deployment and management to ensure every system or product development with project management conforms to the responsibility of providing quality service(s) to the users or stakeholders. The concept of reliability arises due to the need to continually facilitate quality assurance of both products and services. Reliability engineering ensures that some basic fundamental engineering principles and techniques are integrated not only into software development but also towards ensuring effectiveness and efficiency in product and service during project management.

Scholarly articles from different schools of thought defined reliability engineering as the ability of a product to be totally free from fault or failure, also as the ability to minimize failures, faults, or errors in a product or service [1, 7, 9, 14-19]. Faults, error or failure signifies unsuccessful execution of a process in a program, as noted in [14]. The effect of fault occurrence in a software system can be very devastating due to the enormous reliance on humans to these systems. This software system constitutes our daily lives and users depend heavily on them for security, industrial and home automation hence adequate measures are required to guarantee their functionality.

To further expatiate on the necessity of tailoring the principles and techniques of software development methodologies and quality assurance in service delivery through reliability engineering of product, this research paper evaluates previous literature in a bid to profound a basis for studies towards developmental approach, and project execution for cost-effectiveness and quality assurance of software products and services. Software development methodologies serve as a guide towards successful execution of software products and hence should be continuously tailored according to [20-21] to meet the needs of the phenomenon in technological dynamics.

II. RELATED WORKS

The prospect of reliability with respect to how dependable a system existed since the first development of computer system in the 80's [22-23]. This concept of reliability was necessitated due to the frequency of failures in early computers. Reliability as the name implies, refers to trust. It, therefore, means the ability of a software computing system to function and exist without faults, errors or failures [16]. To achieve reliability in a system, according to [22] it is necessary to design a framework that improves the trust of the users-to-computing system through the design and application of a framework that integrates necessary engineering techniques for improving the functionality of

complex systems. Software systems constitute the majority of the complex systems within our environment hence, according to [24] it is necessary to implement design criteria during the development of software's to ensure its reliability is guaranteed. [25-26] suggests that tailoring software development should be considered during development to meet the ever-rising developmental challenges experienced in software engineering due to changes in many factors i.e. users, machine, and environmental requirements. Tailoring simply refers to the act of modifying a particular developmental model or process to suit current project or environment. Although agile development methods are widely employed or used during development or project execution, it is pertinent also that it should be easily tailored with respect to the project to be executed. Considering the dynamics in software development, according to [27] the concept of reliability development model is necessary to eliminate faults is software systems.

III. Tailoring Methodologies

The process of tailoring software development methodologies according to [28] and [29] is an emerging trend practiced in most industries involved in software development albeit little research that has been carried out in this area. In most developmental organizations, the essence of development is based on a stringent approach, which in most cases fails to address all problems arising due to technological changes. Tailoring is necessitated solely to increase performance and productivity, which ultimately ensures efficiency and effectiveness in the quality of a software product.

Since the conceptualization of software development practice, which lays emphasis on standards and guideline for development, many development practices have consequently shifted massively from traditional development practice towards a lightweight agile based development approach. According to [26], [30] building processes from scratch can be very risky with high overhead, as such, many product development organizations tailors (modifies or change) already existing development methodologies to suit the development process of a new system.

This process sometimes can be time-consuming, and most times does not guarantee the total quality of a software product. Flexibility in software development approach according to [31-32] provides an avenue that the stability of the final software product developed is guaranteed, it also ensures that all appropriate mechanisms that guarantee the coherent final system is produced, despite the potential difference in requirements and specifications. The basic fundamental concept in developing new products and services is to ensure that they are highly reliable in dispersing the essential functional values for which they are created for, irrespective of the environment.

In [33], [29], and [30] it is stated that the agile manifesto is based on the application of fundamental, appropriate tools and techniques towards a highly efficient working product with constant user collaboration throughout the phases or process of development. The justification made in agile manifesto towards integrating appropriate tools and techniques in the provision of a high-quality product, [33] gives room for application of systems engineering processes towards ensuring that the goals and objectives of the agile manifestos are met.

IV. Premise for Reliability

Software reliability has over the decades become a bottleneck in both small, large and complex systems. Unlike hardware systems, designing for the software system is more complex and difficult due to systems functionalities and unending changes in both users and environmental requirements [1]. In [34] in Computerworld, it was stated that in America alone software problems and defects as a result of bugs cost the U.S economy an estimated sum of $59.5 billion yearly, in which half of the cost is borne by the end users and the rest by the software developers and vendors. To ensure the issues relating to inadequacies in software development and functionality, [35] suggest benchmarks and standards that would ensure that software systems meet up with requirements and conforms to specification is enforced.

This process would ensure proper engineering of software products towards maximum functionality. According to [16], [36], and [1] software development sectors and their developmental processes, approach or procedures have greatly improved but still far from the issues in regards to design and development for highly reliable software systems. It was further stated by [1] that issues relating to memory loss, leak, corruption, overflow, and deadlocks are still some of the problems that have persisted over time. Reliability, as represented in figure 1, incorporates many other quality characteristics, hence serves as a more comprehensive quality process which ensures all failures and errors are totally eliminated in a computing system.

According to [19], the processes ensured through design and development for reliability ensures that cost of software development is drastically reduced due to the elimination of future recurring maintenance of a software system. Design and development for reliability according to [15] and [37] has not been fully defined and integrated into most development paradigm.in a bid to improve systems prospects according to functionality [16] the concept of reliability was conceptualized. Design, development and testing for reliability ensure that systems conform to specifications and functions maximally, at all times.

V. Reliability Engineering

In most software developmental methodological approach, the process of testing and validation of functional components in a software system results in delays and also cost high without any adequate assurance of functional stability of the software system processes [38] and [1].

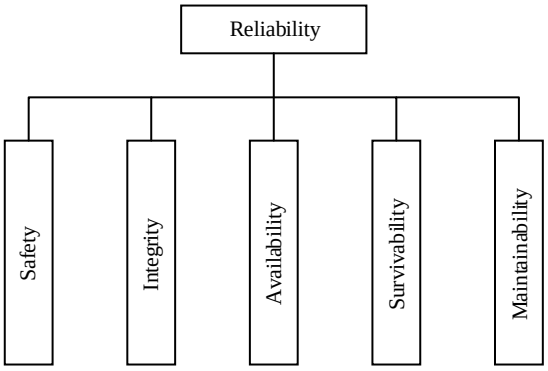

Figure 1: Reliability Attributes [39]

Testing software components in software is a subject of strong conflicting force in software development engineering because most approaches focus more on iteration process, which states that at every level of development, any issues, error, or fault discovered or encountered can be revisited, reevaluated and mitigated. The problem of reiteration is the time spent on revisiting or repairing a single particular system or product. The issues as to frequent problem of a software system as a result of failure or faults in a system according to [1], [14], [16], [26], [35], and [38] is because most software developmental process is not engineered and as such the testing process cannot be adequately relied on and in most cases can cost more than expected due to time of reoccurring maintenance. To enable a competitive and stable software system with less time towards service delivery during development, the fundamental processes of development must be engineered and aligned towards sustainability in products and services.

Reliability engineering uses the operational profile of a software system and guides developers to test software systems and products, in general, more realistically. This process makes it easier for developers to trace the quality level achieved in a system during development or deployment. The process of reliability engineering plays a very important but undervalued role in today's software development process [17]. The study and process of reliability engineering is a very broad discipline which is practiced virtually in most industrial sectors, it ensures quality as a necessity is maintained and improved daily. Reliability engineering which according to [1], [16], [18], [40], and [41] is a process which focuses on basic engineering techniques ensured during design, development, deployment, and maintenance of a system.

[16] states that the ubiquitous nature of most software and its invisible nature makes it more beneficial with respect to space but also harmful. The reason for its negative nature is due to the fact that no one can state when, how, and where a software system would fail, the failure of an important component in a software system can in cure great cost for the users.

Reliability engineering by definition refers to the capability of a product i.e. software, hardware to function at its maximum capacity without any issues arising either from failure or breakdown [1], [16], [40], [42], and [43]. The processor techniques ensured or employed during reliability of product or services ensures it is totally free or minimal defects from faults, errors or failure. The process of reliability is centered upon key attributes towards the operation of failure-free systems within a specified time-frame. This entails that at every given point of time, software systems must function properly and appropriately within its specified duration. Figure 2 depicts the basic processes used in providing a framework towards quality availability in a system as proposed by [14] and [16], it describes the validation process through extensive testing of a software system to ensure that at every developmental phase, all errors, faults or failures are discovered and eliminated.

The reliability framework depicted by [14] and [16] as shown in Figure 1 encompasses four fundamental phases which are reliability objective definition, the operational profiling of the system, modeling and measurement and validation of the reliability.

VI. RELIABILITY ENGINEERING TECHNIQUE

The prospects towards achieving highly reliable software system depend on the application and strict adherence to some fundamental engineering principles, according to [14], [16], [24], [44], and [45], the following are some basic reliability engineering techniques which ensure the provision of high-quality software product:

- Fault avoidance
- Fault detection
- Fault tolerance
- Fault removal

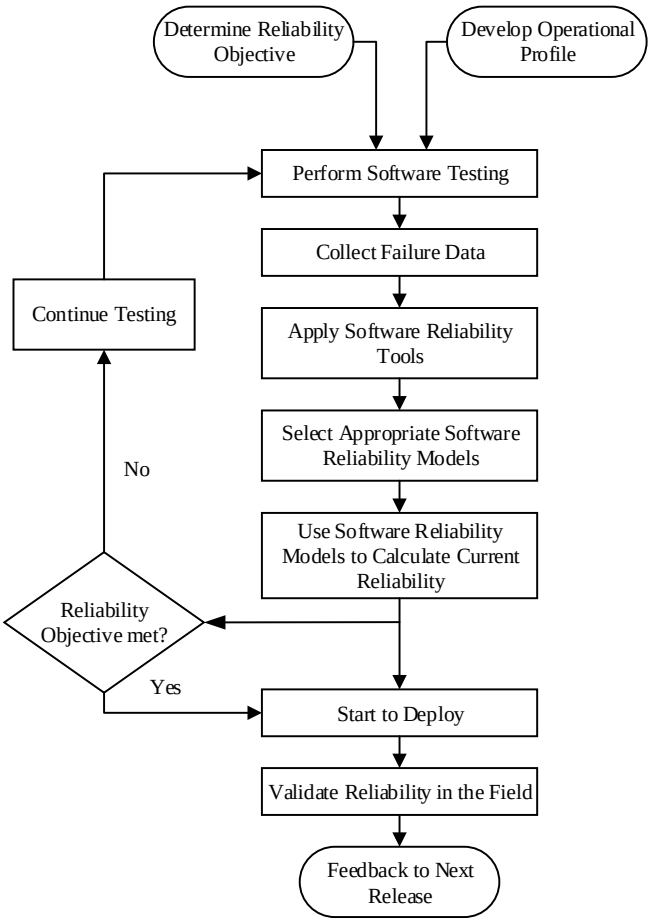

Figure 2: Overview of reliability implementation process [16]

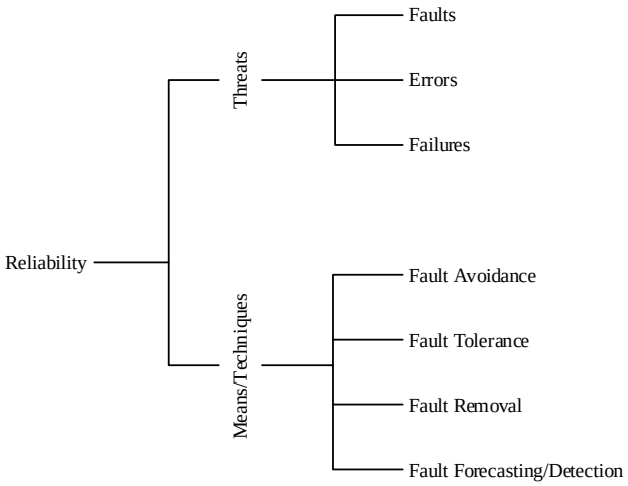

Figure 3: Reliability threats and mitigation techniques [39] and [41]

Fault Avoidance: according to [16] and [24] this technique describes the defensive mechanism ensured towards unreliability, a fault never created would cost nothing to fix. The process in avoidance serves as a foundation towards every developmental methodology, fault avoidance entails that functional requirements i.e. user requirements, systems requirement and the operational environmental requirement is properly collected and critically analyzed to ensure that proper refinement of the systems' functionality is ensured and also defined. This process does not totally guarantee defect-free software but ensures minimal defects which can be further mitigated through the fault removal technique. Some processes ensured in this technique includes information-hiding and strict adherence to all developmental principles because according to [27] poor requirement configuration and requirement elicitation comprise of the main process by which faults are introduced in a system.

Fault Detection: this technique categorically aims at detecting all faults or errors, once the software system is developed and operational, this process recognizes the inadequacies in software systems, as such ensures measurement between the period and time of occurrence and the time the fault is detected. According to [16], if the time period between discoveries of the fault is shorter, it means the software recovery time is better. To achieve maximum detection of faulty components in a software system, proper reliability tools such as root cause analysis and Fault-Tree Analysis (FTA) needs to be deployed to monitor the system concurrently while it functions in order to promptly identify inconsistencies during operation.

Fault Tolerance: this technique according to [46] adds some special features and functions to a software system or program to ensure the system functions properly as well as operates normally irrespective of any triggered defect(s) tolerance according to [47] provides redundancy and stringent adherence and compliance towards system requirements. Error or fault management is the capability of a software system to function properly at all time. According to [48] all systems should be equipped with mechanisms for preventing, detecting, and correcting faults in a bid to reduce the risk of disruption in service during usage. This ensures that processes and tools are employed to reduce excesses as well as eliminate irrelevant content that might create overheads.

In [27], it is suggested that safe modes are integrated into a software system to ensure they perform maximally during the process of fixing detected errors or faults. Fault removal: removal entails that faulty components discovered as a result of reliability testing are replaced by an off-the-shelve method. Proper isolation is necessitated to ensure systems function well and not all faults or errors migrates into larger faults. Reliability tools are utilized to enable total elimination of faults and its effects. According to [16] fault-masking technique hides the effects of errors or failures through redundancy. Reliability testing through tools according to [23] and [49] is an important process in fault removal which ensures extensive testing and evaluation of previously tested components and mitigates all recorded errors or faults.

Table 1
Reliability threats and mitigation techniques [39], [41]

Techniques	Execute
Fault Avoidance	Design Review. Requirement Review. Perform Root Cause Analysis.
Fault Removal	Application Routing Auditing/Checks. Overload Control. Component Application Check.
Fault Tolerance	Application Error/Fault Inline Self-Correction. Perform Audits such as: i. Check for Inconsistencies. ii. Check for Stuck Conditions. iii. Reduce Application Coupling. iv. Separate Coupled Application for Different Functions. v. Constant Application Backup
Fault Forecasting/Detection	Application Fault Localization. Root Cause Analysis. Application Fault Repairs. Update Application Constantly.

VII. Summary and Conclusion

The process of tailoring software development approach has been used since decades either knowing or unknowingly, this process ensures that processes used in development are modified based on the specific project to be executed, the need to provide systems that would serve its specific purpose without interruption or disruption is still a topic for contention, reliability engineering as discussed in this research paper is proposed as an effective mechanism towards quality assurance of software systems. Most organizations that are into software development subscribe to agile developmental approach and manifesto which ensures and proposes iterations during development.

Iterations alone as discussed in this research cannot solve or mitigate software issues completely although software development process can be enhanced based on an agile manifesto which suggests application of tools and techniques to improve on quality during development as well as quality assurance of systems. An overview on the necessity of reliability engineering is also discussed in this paper, its processes and some important technique is also discussed to give an insight towards the importance of engineering software products and services to withstand obstacles irrespective of environmental application.

References

[1] M.I. Malkawi, "The art of software systems development: Reliability; Availability; Maintain," *Performance (RAMP)*, 2013, pp. 1–17.

[2] B. Edson, B. Hansen, and P. Larter, "Software Reliability, Availability, and Maintainability Engineering System (SOFT-RAMES)," *Reliability and Maintainability Symposium 1996 Proceedings International Symposium on Product Quality and Integrity Annual*, 1996, pp. 306–311.

[3] S.D. Carter, and D.M. Deans, "Reliability engineering as a practical application to improving system performance - From concept to system retirement," *Proceedings - Annual Reliability and Maintainability Symposium*, 2011.

[4] C. Lindholm, and M. Host, "Introducing usability testing in the risk management process in software development," *2013 5th International Workshop on Software Engineering in Health Care (SEHC)*, 2013.

[5] N. Zeni, and L. Mich, "Usability issues for systems supporting requirements extraction from legal documents," *2014. 2014 IEEE 7th International Workshop on Requirements Engineering and Law, RELAW 2014 – Proceedings*, 2014.

[6] R. Sattiraju, and H. D. Schotten, "Reliability Modeling, Analysis and Prediction of Wireless Mobile Communications," 2014, pp. 14–19.

[7] G. Kaur, and K. Bahl, "Software Reliability, Metrics, Reliability Improvement Using Agile Process," *International Journal of Innovative Science, Engineering & Technology*, 2014, pp. 143–147.

[8] M. Nkosi, and F. Mekuria, "Improving the capacity, reliability life of mobile devices with Cloud Computing," *2011 IST-Africa Conference, IST 2011, May 11, 2011 - May 13, 2011. 2011 IST-Africa Conference Proceedings*, 2011, pp. 1–9.

[9] B. B. P. Rao, P. Saluia, N. Sharma, A. Mittal, and S. V. Sharma, "Cloud computing for Internet of Things & sensing based applications," *Sensing Technology (ICST), 2012 Sixth International Conference on. [Online]*, 2012, pp. 374–380.

[10] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, 2013.

[11] A.M. Mzahm, M.S. Ahmad, and Y. C. Alicia, "Agents of Things (AoT)," 2013, pp. 159–164.

[12] O. Kodym, F. Benesi, and J. Svubi, "EPe Application Framework in the context of Internet of Things," 2015, pp. 214–219.

[13] R. D. Sriram, and A. Sheth, "Internet of Things Perspectives," *IT Professional*, 2015.

[14] M.R. Lyu, "Design, testing, and evaluation techniques for software reliability engineering," *Euromicro Conference, 1998. Proceedings. 24th.* 1998, pp. XXXIX –XXLVI vol.2.

[15] B. Littlewood, and L. Strigini, "Software reliability and dependability," *Proceedings of the conference on the future of Software engineering*, 2000.

[16] M.R. Lyu, "Software Reliability Engineering: A Roadmap," *Future of Software Engineering (FOSE '07)*, 2007.

[17] M. Houtermans, "Reliability Engineering & Data Collection. Systems," 2007.

[18] D. Raheja, and L. Gullo, "Design for reliability; Wiley-Blackwell," 2012.

[19] M. Silverman, A. Kleyner, "What is design for reliability and what is not?" *2012 Proceedings Annual Reliability and Maintainability Symposium*, 2012.

[20] F. Dai, and T. Li, "Tailoring Software Evolution Process," *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, 2007.

[21] N. A. Zakaria, S. Ibrahim, and M. N. Mahrin, "The state of the art and issues in software process tailoring," *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)*, 2015.

[22] Y. Lin, D. Li, C. Liu, and R. Kang, "Framework design for reliability engineering of complex systems," 2014, pp. 19–24.

[23] A. Pasquini, G. Pistolesi, S. Risuleo, A. Rizzo, and V. Veneziano, "Reliability analysis of systems based on software and human resources," *Proceedings the Eighth International Symposium on Software Reliability Engineering*, 2001.

[24] F. Zeng, and S. Yang, "Design Criteria Development for Software Reliability," *2012 Second International Conference on Intelligent System Design and Engineering Application*, 2012.

[25] P. Xu, and B. Ramesh, "Software Process Tailoring: An Empirical Investigation," *Journal of Management Information Systems. 24 (2)*, 2007, pp. 293–328.

[26] P. Xu, and B. Ramesh, "Using process tailoring to manage software development challenges," *IT Professional. 10 (4)*, 2008, pp. 39–45.

[27] E. Valido-Cabrera, "Software reliability methods," 2006.

[28] R. Akbar, S. Safdar, M.F. Hassan, and A. Abdullah, "Software development process tailoring for small and medium sized companies," *2014 International Conference on Computer and Information Sciences (ICCOINS)*, 2014.

[29] A.S. Campanelli, and F.S. Parreiras, "Agile methods tailoring - A systematic literature review," *Journal of Systems and Software*, 2015.

[30] P. Serrador, and J. K. Pinto, "Does Agile work? - A quantitative analysis of agile project success," *International Journal of Project Management*, 2015.

[31] D. Balasubramaniam, R. Morrison, R.M. Greenwood, and B. Warboys, "Flexible Software Development: From Software Architecture to Process," *The Working IEEE/IFIP Conference on Software Architecture, WICSA '07*, 2007.

[32] A.Q. Gill, "Agile enterprise architecture modelling: Evaluating the applicability and integration of six modelling standards," *Information and Software Technology*, 2015.

[33] I. Inayat, S. S. Salim, S. Marczak, and M. Daneva, S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Computers in Human Behavior*, 2015.

[34] P. Thibodeau "Study: Buggy software costs users, vendors nearly $60B annually," 2012.

[35] W. Li, Y. Yang, J. Chen, and D. Yuan, "A cost-effective mechanism for cloud data reliability management based on proactive replica checking," *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, 2012 pp. 564–571.

[36] N. F. Schneidewind, "Tutorial on Hardware and Software Reliability, Maintainability, and. In: Computer, Network, Software, and Hardware Engineering with Applications," *John Wiley & Sons, Inc.*, 2012 pp. 443–465.

[37] R. Plösch, A. Mayr, and C. Korner, "Collecting Quality Requirements Using Quality Models and Goals," *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference*, 2010.

[38] J.D. Musa, "Introduction to software reliability engineering and testing," *Proceedings the Eighth International Symposium on Software Reliability Engineering - Case Studies*, 1997, pp. 3–12.

[39] L. Algirdas, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, 2004.

[40] P. Kunkun, and L. Xiangong, "Reliability Evaluation of Coal Mine Internet of Things," *2014 International Conference on Identification, Information and Knowledge in the Internet of Things*, 2014.

[41] T. Frühwirth, L. Krammer, and W. Kastner, "Dependability demands and state of the art in the internet of things," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA. 2015-October*, 2015.

[42] S.S. Gokhale, "Software Application Design Based On Architecture, Reliability and Cost," 2004, pp. 1098–1103.

[43] L. Yong-Fei,and T. Li-Qin, "Comprehensive Evaluation Method of Reliability of Internet of Things," *2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2014.

[44] J. Shao, "New thinking and methodologies on reliability engineering," *Proceedings of 2009 8th International Conference on Reliability, Maintainability and Safety, ICRMS 2009. (3)*, 2009, pp. 149–153.

[45] O. Patrick, and A. Kleyner, "Practical Reliability Engineering. 5th Ed. Wiley," 2012.

[46] S. J. Huang, W. C. Chen, and P. Y. Chiu, "Evaluation Process Model of the Software Product Quality Levels," *2015 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration*, 2015.

[47] J. Yang, Y. Liu, M. Xie, and M. Zhao, "Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes," *Journal of Systems and Software*, 2016.

[48] V. Nassar, "Common criteria for usability review," *Work. 41 (SUPPL.1)*, 2012, pp. 1053–1057.

[49] H. Koziolek, B. Schlich, and C. Bilich, "A Large-Scale Industrial Case Study on Architecture-Based Software Reliability Analysis," *ISSRE'21st*, 2010, pp. 279–288.