JOIV : Int. J. Inform. Visualization, 8(4) - December 2024 2344-2353

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

A Systematic Literature Review on Characteristics Influencing Software Reliability

Lehka Subramanium [a,*], Saadah Hassan [a], Mohd. Hafeez Osman [a], Hazura Zulzalil [a]

[a] *Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Selangor, Malaysia*

Corresponding author: [*]GS59140@student.upm.edu.my

***Abstract*—Reliability, as a non-functional requirement, is a crucial aspect that refers to the system's ability to perform its intended functions consistently and without failure over an extended period. It is essential in designing and implementing software systems, as it affects software quality. Maintaining software reliability is a significant challenge, as it is directly impacted by factors such as the complexity of the software design, the amount of code, and the measures taken to secure the system from unauthorized use. There are significant growing appeals for predicting reliability to account for risks. Research on reliability risk assessment has a long tradition; unfortunately, comprehensible reliability characteristics are still vague when determining potential risks. Clearly defining, prioritizing, and addressing reliability characteristics is essential for delivering reliable, high-quality software that meets user needs and business goals. The ignorance and lack of comprehensive reliability characteristics have evolved into inaccurate risk assessment, triggering malfunctions in the operational environment. Comprehensive characteristics are key elements to predict and estimate software reliability. The reliability characteristics could determine the precise objective of reliability efforts. This systematic literature review aims to identify the key characteristics influencing software reliability, the potential risks associated with these characteristics, and the metrics used to measure and assess them. Thirty-one research articles related to research questions have been reviewed. The findings indicate that comprehensive reliability characteristics could identify, classify, and prioritize potential risks, improving current metrics. It can be concluded that the accurate potential reliability risk can demonstrate the consequence of failure.**

***Keywords*—Reliability; characteristics; risk; metrics; systematic literature review.**

Manuscript received 11 Dec. 2023; revised 17 Apr. 2024; accepted 25 Oct. 2024. Date of publication 31 Dec. 2024.

I. Introduction

Recent research has highlighted the challenges of predicting software reliability as a non-functional requirement due to varying workloads, user behaviors, and operational conditions [1]. Traditional models often fail to capture the complexity of modern systems, leading to unreliable predictions [2]. Poor reliability predictions can result in unexpected downtimes, unplanned maintenance, and costly post-deployment fixes. On the other hand, reliability models require extensive failure data for accurate predictions, but gathering this data, especially in the early stages of development, is often difficult. Insufficient data leads to unreliable models and less confidence in reliability estimates [3]. Furthermore, many organizations lack robust methodologies for assessing risks related to reliability during early design and development stages. This results in unforeseen issues during later stages, including production [4].

The lack of well-defined reliability characteristics is a root cause that exacerbates reliability problems across multiple stages of software development and operations. Organizations cannot effectively identify and mitigate risks without clearly understanding the specific reliability characteristics critical to the system. This leads to unforeseen issues during operations, particularly under stressful conditions [2]. This is a complex challenge, and simplifying it requires transparent reliability characteristics to synthesize suitable applications, methodologies, and risk assessment [2]. For instance, in safety-critical applications, software reliability is paramount. Attributes like fault tolerance and robustness are crucial to prevent catastrophic failures that could lead to injury or loss of life. Without a shared understanding of "reliable" software, developers, testers, and stakeholders may have different interpretations, leading to confusion and misaligned priorities [3].

Furthermore, if the characteristics are not well defined, measuring them objectively and tracking progress toward

achieving them becomes challenging. The challenges make it difficult to assess the software's reliability risk and identify areas for improvement [5]. Consequently, a lack of evident reliability characteristics makes it difficult to establish consistent metrics and monitoring practices [6], [7]. Different teams might focus on different metrics, leading to fragmented and potentially misleading views of reliability. Relatively, with clear reliability characteristics and accurate risk forecasts, these metrics can measure resource allocation, testing strategies, and release readiness decisions [3]. Therefore, developers can monitor the effectiveness of their efforts to improve software quality [8].

A Systematic Literature Review (SLR) is conducted in response to these challenges to provide a comprehensive overview of reliability characteristics, associated risks, and relevant metrics. The contributions of this work include (i) a classification of reliability-related quality characteristics, (ii) the identification and characterization of potential reliability risks as described in system engineering, and (iii) the identification of existing metrics for concrete measurement of these characteristics.

The structure of this paper is as follows: Section II presents the motivation of this research; Section III discusses the related works. Section IV discusses the methodology protocols. Consequently, section V presents the findings and discussions, and section VI illustrates the conclusion. Many quality models recognize reliability as a key aspect of overall software quality [9], [4], [5], [6]. Despite continuous research and adherence to existing standards, challenges in ensuring software reliability persist, often due to inadequately defined reliability characteristics [7], [10]. When these characteristics are not clearly articulated, controlling potential risks becomes difficult, leading to negative effects.

Thus, this study aims to thoroughly investigate software reliability characteristics, associated risks, and relevant metrics. The contributions of this research have significant implications for ongoing reliability efforts across the industry. The following key reasons drive the motivation for this work:

a. Enhanced System Robustness: Understanding reliability characteristics helps identify strengths, weaknesses, and vulnerabilities, facilitating the development of more robust and resilient systems.
b. Addressing Emerging Challenges: New reliability challenges arise as software systems evolve and grow more complex. Research in this area is crucial for devising innovative solutions and techniques tailored to specific characteristics, ensuring the reliability of next-generation software.
c. Advancing Software Engineering: Investigating software reliability characteristics advances the field of software engineering by deepening our understanding of metrics essential for creating dependable software systems.

Investing in this research will enable developing more reliable and trustworthy software systems better equipped to meet modern applications' increasing demands and complexities.

II. Materials and Method

There have been numerous studies to investigate the significance of reliability engineering [11], [12], reliability metrics [2], risk assessments [13], [14], reliability prediction and evaluation [3], [4], [15], [16], [12] and reliability assurance [17], [18]. In the context of this literature review, a few papers that do not present novel approaches will be discussed, yet the papers are relevant in this scope of reliability. Existing fundamental works on reliability provide the foundational concepts of reliability in quality models [9], [19], [20]. Hovorushchenko [9] evaluates various software quality models, including the McCall, Boehm, Dromey, FURPS, ISO 9126, ISO 25010, Bertoa, and GEQUAMO models. ISO 25010 is the most comprehensive quality model compared to others [20]. In this research, we have used the ISO 25010 quality model to classify the quality attributes.

Many studies have been recorded on the characteristics of quality attributes [21], [9], [22], [23], [24]. Although different studies yielded varying reliability characteristics, the common aim is to determine the pattern of reliability characteristics. From the literature review, we discovered that the significant characteristics are recoverability, maturity, fault tolerance, and availability. Each characteristic plays a distinct role in addressing reliability challenges through targeted risk assessment techniques. The significant reliability characteristics could identify potential challenges, anticipate failures, and implement appropriate migration strategies at different stages. Richard et al. [13] induced the relationship between reliability and probabilistic risk assessment. Next, Zhang proposed a field product reliability risk assessment by initiating a quantitative risk assessment for field products [25]. On the other hand, Neha and Kirti emphasize the complexity of estimating reliability in Service-Oriented Architectures due to their interconnected nature. The authors identify four key factors significantly influencing SOA reliability [26]. Likewise, Yacoub et al. [9] made a seminal contribution by proposing a methodology for assessing reliability risk at the architectural level, which is the early phases of the development lifecycle using complexity factors.

While numerous studies have proposed risk assessment techniques for specific domains like power grids, component-based systems, and even for different phases of the software development life cycle (SDLC), there is a noticeable gap in research that explicitly aligns risk assessment with key reliability characteristics. Despite the broad range of existing approaches, few studies delve into how specific reliability characteristics contribute to comprehensive risk assessment across all SDLC phases. For instance, while fault tolerance is critical in high-availability systems, there is limited guidance on assessing risks associated with insufficient fault tolerance [16]. Recoverability, maturity, and availability are similarly underrepresented in risk assessment models. Current methods often treat these characteristics as secondary considerations rather than focal points, leading to assessments that may not fully capture the risks impacting overall reliability.

The relationship between metrics and reliability risk assessment is symbiotic. Metrics provide the quantitative foundation for precise, data-driven risk assessments, while effective risk assessment models rely on relevant and accurate metrics to identify, prioritize, and mitigate potential threats to system reliability [25]. Together, they enable a structured approach to achieve and maintain high levels of reliability in software systems [4]. Baybulatov and Promyslov [5] discussed an Availability Risk Assessment metric. They used

industrial automation and control systems (IACS) security as the basement to evaluate the risk. Apart from this, research in 2010 by Quyoum et al. [27] listed fundamental metrics based on product, process, project management, and fault and failure metrics tailored to reliability as the baseline reference.

When risk assessment models do not prioritize these reliability characteristics, the metrics derived from such assessments may fail to measure a system's true risks accurately. For instance, a metric designed to measure fault tolerance might not consider the full range of failure scenarios, leading to an overly optimistic view of system reliability. In conclusion, there is a clear need for research that combines reliability characteristics, risks, and metrics in a structured and integrated manner. Such research will result in optimum reliability research efforts applied across diverse software systems, providing actionable metrics that drive better reliability outcomes [6].

A comprehensive Systematic Literature Review (SLR) methodology is conducted to thoughtfully and concretely address the formulated questions [10]. The sequential phases of the SLR protocol are establishing research questions, database selection, defining search strategy, determining inclusion and exclusion criteria, evaluating quality criteria, snowballing, and doing metadata analysis. The processes that make up our SLR design are depicted in Fig 1.

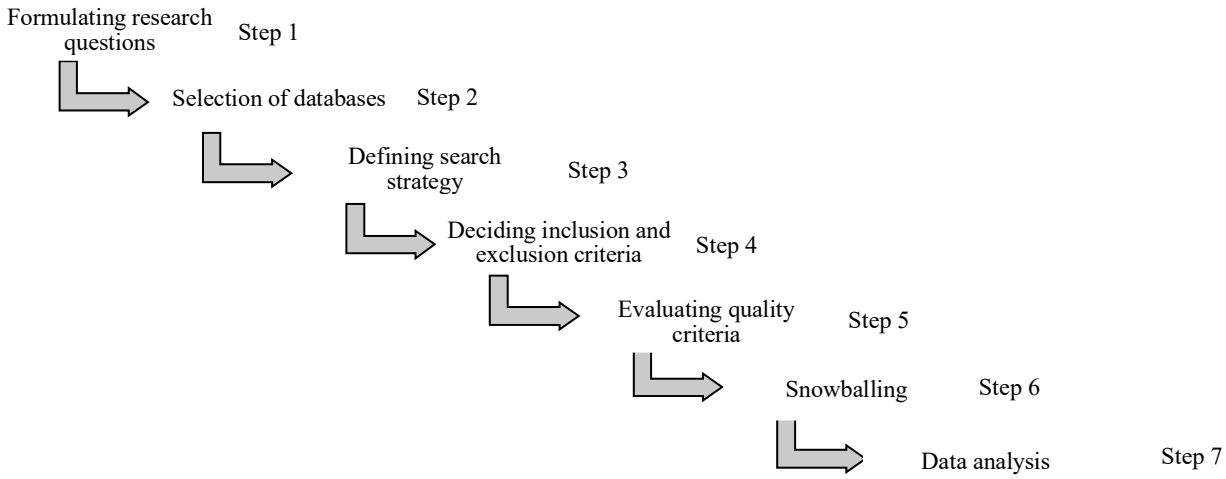

Fig. 1 SLR Design Procedure

A. Research Questions

Research questions (RQ) help organize the collection of references, define study parameters, and direct the development of the research procedure. The RQs are shown in Table I to construct the search query string.

TABLE I
RESEARCH QUESTIONS AND REASONS

ID	Research Questions	Reasons
RQ-1	What are the essential characteristics that determine system reliability?	This question aims to identify and analyze the key characteristics that influence and determine the reliability of a system.
Q-2	What are the potential risks linked to each reliability characteristic in a system?	Identify and analyze the risks associated with each reliability characteristic in a system to better understand and mitigate potential reliability issues.
RQ-3	What reliability metrics are currently used in systems?	To identify and review the reliability metrics currently utilized in systems to assess their effectiveness and applicability.

B. Selection of databases

Data collection uses the following databases and libraries, encompassing a broad spectrum of topics relevant to the research area. These libraries are selected for their strong relevance and robust, user-friendly search engines, making them well-suited for comprehensive and automated searches. Table II provides a list of these libraries.

TABLE II
ONLINE DATA SOURCES

Database sources	Website
IEEE	https://ieeexplore.ieee.org/Xplore/home.jsp
Scopus	https://www.scopus.com/search/form.uri?display=basic#basic
Science Direct	https://www.sciencedirect.com/
Springer	https://link.springer.com/
Wiley Online Library	https://onlinelibrary.wiley.com/

C. Search Strategy

Our objective is to carefully structure our search strategy, which is fundamental to any research project. The initial step in this phase of the SLR protocol involves combining keywords to develop an effective search string. Simply using individual keywords is inadequate; instead, they must be combined in various ways to create a search string suitable for different journals and digital libraries. The search strategy consists of four steps: defining keywords, constructing the search string, selecting relevant databases, and executing the search process. This approach is inspired by the methodology proposed by Kitchenham [28].

1) Defining Keyword: The search terms are defined to retrieve the most relevant results of papers. Table III displays the list of all the keywords created for searching purposes.

TABLE III
RESEARCH QUESTIONS WITH KEYWORDS

Research Questions	Keywords
What are the essential characteristics that determine system reliability?	(character* OR attribute OR feature OR factor) AND (reliab*) OR (quality model)
What are the potential risks linked to each reliability characteristic in a system?	(risk OR threat OR assess* OR challenges) AND (reliable*)
What reliability metrics are currently used in systems?	(metric OR technique OR standard OR bench*) AND (reliable*)

2) Search String: The keywords for each question are combined to create a search string. The search query is examined across many sources and adjusted until the most pertinent results are found.

a. Major words derived from the topic and research questions
b. Finding synonyms or alternative spellings for important phrases
c. Identification of keywords
d. The Boolean operator OR is used for synonyms or other spellings
e. Linking Boolean AND operator with essential terms.

Pilot searches are conducted to refine our search strategy and optimize results. Our search string is structured into two parts, focusing on characteristics, risks, and metrics and specifically on reliability.

3) The search query was conducted in April 2024, utilizing automated and manual methods to identify relevant studies. According to Kitchenham [28], computerized searches are more effective than manual searches. A manual search was also performed to validate the search string. The search string detailed above was applied to each database, as outlined in Table IV. The results included 139 studies from IEEE Xplore, 51 from ScienceDirect, 48 from Scopus, 23 from Wiley, and 20 from Springer.

A careful examination of the chosen papers is conducted regarding the inclusion and exclusion criteria listed in Table IV.

TABLE IV
INCLUSION AND EXCLUSION CRITERIA

Inclusion criteria
Research papers published in English
Research papers published from 2002 to 2024
Scholarly papers published in computer science and software engineering journals or conferences
Exclusion criteria
Papers written in other than English are not included
Duplicate papers are eliminated
Research papers with fewer than three pages are excluded
Studies missing abstracts are excluded
Excluded books, thesis, editorials, prefaces, article summaries, interviews, news, reviews, correspondence, discussions, comments, reader's letters, and summaries of tutorials, workshops, panels, and poster sessions. Research papers excluding reliability as a quality attribute dropped.

Initially, duplicate articles are eliminated, and each one is then compared to the established keywords and developed research inquiries. Papers that did not offer thorough responses to the questions are disqualified. Next, each paper is evaluated using inclusion-exclusion criteria based on the title, abstract, and complete reading. Peer-reviewed journal articles, conference proceedings, book chapters, editorials, and magazine articles are selected for inclusion in the study. When several versions of the same article exist, the most recent, comprehensive, and updated copy is selected for inclusion in the research, and the other copies are eliminated. Through conflict analysis, bias is prevented at every level of the selection process.

D. Quality Criteria

After journal papers pass the screening phase, a quality assessment is conducted to evaluate their relevance and viability. The quality assessment process for this research involves answering specific evaluation criteria with either "Yes" or "No." A "Yes" indicates that the paper is suitable for inclusion, while a "No" signifies that it does not meet the required criteria, as shown in Figure 2.

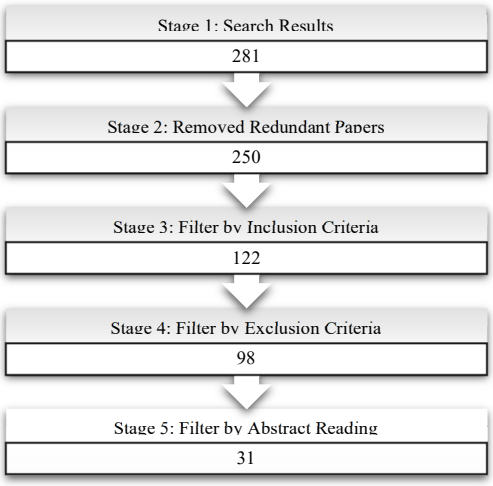

Fig. 2 Quality assessment

E. Snowballing

Snowballing is an essential research technique that builds on relevant studies to discover additional research [44]. We performed both forward and backward snowballing for key outcomes in our study. The process involved several steps: initially, 68 papers were identified. In the next step, after reviewing the titles, 45 papers were selected. Further refinement by reading the keywords and abstracts narrowed the selection to 22 papers. Finally, after a thorough review, the list was reduced to 5 papers as shown in Figure 3.

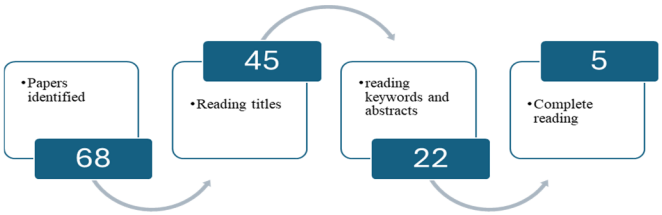

Fig. 3 Snowballing

F. Data Analysis

This stage involves detailing the findings of the selected research papers, which will be analyzed by addressing the predetermined research questions.

TABLE V
RELIABILITY CHARACTERISTICS

Software Quality Model	Characteristics	Paper	Quality Attributes
McCall	Accuracy	[9], [24]	
	Error Tolerance	[9], [24]	
Boehm	Self-Contentedness	[9]	
	Integrity	[9]	
	Accuracy	[9]	
FURPS	Frequency And Severity of Failures	[9]	
	Recovery To Failures	[9]	
	Time Among Failures	[9]	
ISO 9126	Maturity	[9], [22], [21]	
	Fault Tolerance	[9], [22], [21]	
	Recoverability	[9], [22], [21]	
	Reliability Compliance	[9], [22], [21]	
ISO 25010	Maturity	[9], [18], [29]	Reliability
	Availability	[9], [18],[19]	
	Fault Tolerance	[9], [18]	
	Recoverability	[9], [18]	
Bertoa	Maturity	[9],	
	Suitability	[9]	
Alvaro	Recoverability	[9]	
	Fault Tolerance	[9]	
	Maturity.	[9]	
Rawashdeh	Recoverability	[9]	
	Maturity	[9],	
SQO-OSS model	Maturity	[9]	
	Effectiveness.	[9]	

III. RESULTS AND DISCUSSION

The systematic mapping study yielded several insights related to the research issues. The findings are summarized as follows:

A. RQ1: What are the essential characteristics that determine system reliability?

Based on Table V [7], reliability is consistently recognized as a fundamental quality attribute across all major quality models. This underscores its critical role in ensuring consistent system performance and user satisfaction across various frameworks and standards. Reliability is not a monolithic concept but is composed of several individual characteristics.

Among the various reliability characteristics, the following are most frequently mentioned in quality models:

1) Availability: This measure measures the proportion of time a system is operational and accessible when required. It is crucial for ensuring consistent system performance and minimizing downtime.

2) Error Tolerance: This measure reflects the system's ability to handle and manage errors without complete failure. It is vital for maintaining functionality despite errors or faults.

*3) Recoverability***:** Assesses the system's capacity to recover from failures and restore regular operation. It is essential to minimize the impact of failures and ensure rapid recovery.

*4) Maturity***:** Maturity indicates the system's stability and robustness, often reflecting its history of usage and evolution. It impacts long-term reliability and performance.

Integrating these key characteristics into a cohesive strategy is essential to effectively managing reliability [22]. Understanding how availability, error tolerance, recoverability, maturity, and other factors interact helps develop comprehensive risk assessments and reliability metrics.

B. RQ2: What are the potential risks linked to each reliability characteristic in a system?

Table VI shows the grouping of potential risks specific to the reliability characteristics. Identifying potential risks tied to reliability characteristics is crucial for the following factors:

1) Anticipating Challenges: By foreseeing reliability risk, you can design mitigation strategies early in development.

2) Prioritize Critical Areas: Some risks may have a more significant impact. Understanding these allows focused resource allocation for testing and improvements

3) Improve Component Selection: By clearly identifying potential risks, you can make informed decisions about component integration into domains like Software Product Line (SPL) and Component-Based Software Engineering (CBSE).

Research Question 2 (RQ2) revealed that each characteristic has individual potential risks regardless of the phases of the software. Each reliability risk can be analyzed to decide its likelihood and potential impact. With this, strategies can be developed to minimize high-impact risks. Regularly identifying and analyzing potential reliability risks allows for continuous improvement of the software development process. This iterative approach helps refine practices and deliver increasingly reliable software over time.

TABLE VI
POTENTIAL RISK

Characteristics	Potential risk	Description	Paper
Accuracy	Incorrect data precision tolerance	Impacts the precision of computations, leading to inaccurate results.	[30],[23],[31],[32], [11]
	Data collection failure	Incomplete or erroneous data can lead to inaccurate system outputs.	
	Hardware processing capability	Limitations in hardware performance can result in incorrect calculations, affecting accuracy.	
	Incorrect value in the specification	Errors in the specification can lead to incorrect implementation and inaccurate outcomes.	
	Completeness of specification	Missing or incomplete specifications can cause the system to behave inaccurately.	
	Incorrect flow of task execution	Deviations in task execution can lead to incorrect results.	
	Deficiencies in data structures	Improperly structured data can lead to inaccuracies in processing and output.	
	Flaws in reliability policies	Inadequate policies can lead to conditions where accuracy is compromised.	
	Specific points and critical values in procedure;	Errors at key steps or critical points can result in inaccurate outputs.	
	Errors from software transplantation modifications	Changes during software migration can introduce inaccuracies.	
	Approximation of the algorithm leads to the imprecise variable region;	Inaccurate algorithm approximations can result in errors and imprecise outcomes.	
Error Tolerance / Fault tolerance	Inaccurate response time for tolerance	If the system cannot tolerate delays accurately, it may fail to handle errors gracefully.	[18],[13],[16], [33],[34],[35], [36]
	Inaccurate mean time to failure	Affects the ability to predict and handle failures effectively, leading to reduced fault tolerance.	
	Incorrect boundary value tolerance	Defining acceptable value ranges can lead to improper handling of deviations and reduce error tolerance.	
	Redundancy of components	adequate or excessive redundancy can fail to provide fault tolerance or lead to unnecessary complexity.	
	Mean fault notification time	Delays in fault detection can hinder prompt responses, reducing the effectiveness of fault tolerance mechanisms.	
	Failure avoidance	Insufficient strategies to avoid failures impact both fault tolerance and error tolerance.	
	Fault tolerance mechanism	Inefficient or poorly implemented mechanisms can reduce the system's capacity to recover from faults.	
Integrity	Consistency of data in the specification	Inconsistencies in data specifications can lead to incorrect data handling and compromise data integrity.	[10], [26]
	Adherence to the needs	Failure to meet specified requirements can result in data integrity issues, as the system may not handle data according to the expected standards.	
	Weak authentication of data storage	Insufficient authentication measures can lead to unauthorized access and potential tampering with stored data, affecting data integrity.	
	Leakage of data loss/corruption	Loss or corruption of data compromises its integrity by making it unreliable or invalid.	
	Leakage of data loss/corruption	If the accuracy of data cannot be verified, it undermines the confidence in data integrity.	
	The accuracy of data is non-verifiable		
Maturity	Consistency of data in the specification	Inconsistent specifications can hinder the maturation of the software development process, leading to incomplete or erroneous implementations.	[18],[37],[38], [15]
	Software coding mistake	Errors in coding can affect the software's ability to evolve and mature properly, leading to persistent defects and unreliable performance	
	Adherence to the needs	Failure to align with requirements can delay the maturity process by causing continuous adjustments and rework	
	Weak authentication of data storage	Inadequate security measures can impact the stability and reliability of the software as it matures, potentially exposing it to vulnerabilities.	
	Leakage of data loss/corruption	Persistent issues with data loss or corruption can indicate underlying problems that prevent the software from reaching a mature, stable state.	
	Component wear-out	The degradation of components over time can affect the overall maturity of the system, indicating the need for maintenance or upgrades.	
	Manufacturing imperfection	Imperfections in the manufacturing process can lead to defects that affect the reliability and maturity of the software.	
	Incorrect MTBF	Misestimation of MTBF can mislead reliability assessments, affecting the perceived maturity of the software.	
	Incorrect fault correction	Ineffective or incorrect fault correction practices can impede the software's progression toward a mature and reliable state	
	Inaccurate failure rate	Misrepresentation of failure rates can lead to incorrect assessments of software maturity and reliability.	
	Less test coverage	Insufficient testing coverage can prevent the identification of defects and issues, delaying the maturity and stability of the software.	
Recoverability	Response intervals captured by software/hardware upon failure	Inadequate or inaccurate capture of response intervals can hinder the ability to assess and improve the recovery process.	[18],[7], [39], [40]
	Lack of resources in the new environment	Insufficient resources or support in a new environment can impede the system's ability to recover effectively from failures.	
	Inability to recover in a new environment	Challenges in adapting recovery procedures to a new environment can lead to prolonged or failed recovery efforts.	

Characteristics	Potential risk	Description	Paper
	Longer recoverability time of an operation	Extended recovery times can negatively impact the overall reliability and performance of the system	
	Mean recovery time	An inaccurate estimation of mean recovery time can lead to unrealistic expectations and inadequate planning for recovery processes.	
	Backup data completeness	Incomplete or outdated backup data can hinder the ability to restore the system fully, affecting overall recoverability	
Availability	System availability	Factors that affect the system's overall availability, such as downtime or system outages, impact its ability to provide continuous service.	[41],[18], [42]
	software replication	Inadequate or incorrect replication processes can lead to reduced availability, as failures in one instance may not be appropriately mitigated.	
	automated detection	Ineffective automated detection of issues can delay response times and impact system availability.	
	failover	Inadequate failover mechanisms can lead to more extended downtime or service interruptions during failures.	
	hot swap	Problems with hot-swapping components can affect the system's ability to maintain availability during component replacements or upgrades.	
	Inaccurate business time in the specification	Incorrect specification of business-critical times can impact system availability during peak periods.	
	Installation capabilities	Insufficient installation capabilities can delay deployment or upgrades, affecting overall system availability.	
	Network	Network issues, such as outages or performance degradation, can impact the system's availability.	
	Natural casualties	Natural disasters like earthquakes or floods can disrupt system operations and reduce availability.	
	human error	Mistakes by operators or administrators can lead to outages or reduced system availability.	
	lack of transparent unavailability	Poor communication about system unavailability can lead to confusion and reduced confidence in system availability.	
	Sufficient bandwidth	Insufficient bandwidth can affect the system’s performance and availability, especially under high-load conditions.	
	Installation effort/experiment	High installation effort or experimental setups can delay deployment and impact availability.	
	Radiation	Radiation exposure can cause hardware malfunctions, affecting system availability.	
	Electromagnetic interference	Interference from electromagnetic sources can disrupt system operations and impact availability.	
	Battle damage	Damage from conflicts or warfare can affect the system’s physical infrastructure and availability.	
	Wireless channel noises	Interference or noise in wireless channels can disrupt communication and impact system availability.	
	Mean down time	Inaccurate estimation of mean downtime can affect planning and management strategies for maintaining availability.	
Suitability	Poor design	Inadequate or ineffective design can lead to a system not meeting user needs or requirements, impacting its suitability for its intended purpose.	[15],[43], [16]
	Module complexity - Structural complexity - Code complexity - Interface complexity	- Complex module structures can make it difficult to understand, maintain, and ensure the module meets its requirements. - Complex code can introduce errors and make it harder to ensure the module functions correctly and meets its intended use. - Complicated interfaces between modules can lead to integration issues and affect the overall suitability of the system	
	Poor component selection	Choosing inappropriate or incompatible components can affect the system's ability to fulfill its intended functions, reducing its suitability for specific requirements or environments.	
Effectiveness	Application user friendliness	Poor user interface design can reduce user interaction efficiency and hinder the software's effective use.	[15],[44], [45]
	Poor requirement changes	Ineffective management of requirement changes can lead to system design and implementation inefficiencies, affecting overall performance.	
	Limitation in power	Insufficient power resources can affect the performance and efficiency of the system, particularly in energy-intensive applications.	
	Processing and communication capacity	Limitations in processing power or communication capacity can lead to bottlenecks and reduced system efficiency.	
	Memory resources	Inadequate memory resources can lead to performance degradation, impacting the efficiency of data handling and processing	
	Module change rate	Frequent changes to modules can introduce inefficiencies due to the need for constant updates and rework, affecting system performance and stability.	

C. RQ3: What reliability metrics are currently used in systems?

Table VII shows the maturity and stability of existing software in assessing software systems' reliability characteristics. These metrics provide quantitative insights into how well a system performs concerning reliability characteristics. They are applied throughout the software lifecycle—from development processes to post-deployment management. By aligning metrics with specific reliability characteristics, you can target critical areas such as accuracy, fault tolerance, and recoverability, ensuring that your software consistently meets reliability expectations

TABLE VII
EXISTING RELIABILITY METRICS

Metrics	Reliability Characteristics	Purpose	Paper
Mean time to failure (MTTF)	Fault tolerance	Measures the system's ability to continue operating despite the presence of faults or failures.	[7],[2], [27]
Defect Removal Efficiency (DRE)	Fault tolerance, Maturity		[42] [4],[46]
Mean time between failure (MTBF)	Fault tolerance, availability	MTBF measures the average time between failures, reflecting the system's stability and maturity	[47], [48] [31]
Rate of occurrence of failure (ROCOF)	Fault tolerance	ROCOF measures how often failures occur over a period, providing insights into how frequently the system encounters faults.	[30] [43], [17], [6]
Test Coverage Metric	Maturity, Fault Tolerance	Evaluates the extent to which the software has been tested, including the number of code paths, branches, or functionalities tested	
Mean time to repair (MTTR) Recovery time	Recoverability, Integrity	Average time taken to restore service after a failure.	
Probability of failure on demand (POFOD)	Suitability, Effectiveness, Error Tolerance	Measures the likelihood that the system will fail when a request is made	
Service Availability (AVAIL)	Availability	The percentage of time the system is operational and available for use.	
Function Point Metric	Accuracy, Suitability	Measures the functionality delivered by the system relative to user requirements	
Fault tree / Failure tree analysis (FTA)	Fault Tolerance, Integrity, Error Tolerance	A technique used to model the failure paths in a system to identify potential risks and failure causes.	
Failure mode and Effect Analysis (FMEA)	Recoverability, Maturity, Error Tolerance	Identification of potential failure modes and their effects on the system.	
Neural Framework	Dynamic Behavior, Self-Contentedness, Fault Tolerance	Using neural network models to assess and predict various aspects of software reliability, such as failure rates, fault tolerance, and performance under different conditions.	
The dynamic complexity of state charts	Suitability, accuracy, fault tolerance	intricacy and variability in the behavior of state charts as the system operates and reacts to different inputs and conditions.	
Dynamic coupling between components	Self-contentedness, integrity, fault tolerance	Degree of interdependence and interaction between different components of the system during runtime.	
Defect Density	Maturity, Accuracy	The proportion of defects removed before release compared to those found after release.	
Test Coverage		Percentage of code, paths, branches, or functionalities tested.	
Failure rate	Accuracy	Identifies how precise and correct the software outputs are under typical conditions.	
Fault Intensity		Number of failures observed per time interval or operation count.	
Failure Severity index	Integrity	Monitors how well the system preserves data integrity and maintains consistent states.	
Growth Model (Goel-Okumuto, Musa)	Fault tolerance, recoverability	Statistical models that predict reliability based on observed failure data.	
Software Reliability index (SRI)	Accuracy, fault tolerance, maturity, suitability	Quantifies the impact of different types of failures based on severity levels.	

IV. CONCLUSION

In a nutshell, this systematic literature review offers a comprehensive understanding of the factors impacting software reliability. Key reliability characteristics, such as accuracy, fault tolerance, recoverability, maturity, integrity, and availability, are critical for ensuring software systems' robustness and reliability. The review highlights potential risks of each characteristic, such as frequent failures, inadequate recovery mechanisms, and inconsistent performance. Furthermore, a detailed analysis of reliability metrics provides insights into how these risks can be measured, monitored, and mitigated, enabling more reliable software design and development.

The review establishes a structured approach for identifying essential reliability characteristics, offering a solid foundation for reliability assessment. Also, by highlighting specific potential risks associated with reliability characteristics, the SLR provides a deeper understanding of common problems that affect software reliability. This insight helps researchers and practitioners recognize critical areas that need attention and improvement. Also, documenting potential risks helps identify gaps in current research and areas where further investigation is needed. This guides future research by pinpointing where existing studies may fall short or where new approaches could be developed.

Future research could focus on developing integrated models that combine the identified characteristics, risks, and metrics into a comprehensive reliability evaluation

framework. As systems become more dynamic and complex, future work could explore adaptive metrics that account for changing operational environments, real-time updates, and evolving user needs. Lastly, with the increasing adoption of microservices, cloud computing, and AI-based systems, future research could investigate how these technologies influence reliability characteristics and adapt metrics accordingly.

ACKNOWLEDGMENT

We want to thank Universiti Putra Malaysia for all the support given.

REFERENCES

[1] L. Subramanium, S. Hassan, H. Zulzalil, and M. H. Osman, "Identification of Emergent Properties Occurrences Factors in System-of-Systems," *2023 IEEE Int. Conf. Comput. ICOCO 2023*, pp. 71–76, 2023, doi: 10.1109/icoco59262.2023.10397923.

[2] V. K. Singh, R. A. Khan, and S. W. Abbas Rizvi, "Revisiting Software Reliability Engineering with Fuzzy Techniques," pp. 1037–1042, 2016.

[3] A. Mohan and S. K. Jha, "Predicting and accessing reliability of components in component based software development," *2019 Int. Conf. Intell. Comput. Control Syst. ICCS 2019*, no. Iciccs, pp. 1110–1114, 2019, doi: 10.1109/iccs45141.2019.9065290.

[4] C. Haritha Madhav and K. S. Vipin Kumar, "A method for predicting software reliability using object oriented design metrics," *2019 Int. Conf. Intell. Comput. Control Syst. ICCS 2019*, no. Iciccs, pp. 679–682, 2019, doi: 10.1109/iccs45141.2019.9065541.

[5] A. A. Baybulatov and G. Promyslov, "A Metric for the IACS Availability Risk Assessment," *Proc. - 2022 Int. Russ. Autom. Conf. RusAutoCon 2022*, pp. 750–754, 2022, doi:10.1109/rusautocon54946.2022.9896250.

[6] A. Jatain and Y. Mehta, "Metrics and models for Software Reliability: A systematic review," *Proc. 2014 Int. Conf. Issues Challenges Intell. Comput. Tech. ICICT 2014*, pp. 210–214, 2014, doi:10.1109/icicict.2014.6781281.

[7] O. Stover, P. Karve, and S. Mahadevan, "Reliability and risk metrics to assess operational adequacy and flexibility of power grids," *Reliab. Eng. Syst. Saf.*, vol. 231, no. November 2022, p. 109018, 2023, doi:10.1016/j.ress.2022.109018.

[8] C. Wang and A. Mosleh, "Qualitative-Quantitative Bayesian Belief Networks for reliability and risk assessment," 2010 Proceedings - Annual Reliability and Maintainability Symposium (RAMS), pp. 1–5, Jan. 2010, doi: 10.1109/rams.2010.5448022.

[9] T. Hovorushchenko, "The software emergent properties and them reflection in the non-functional requirements and quality models," *Proc. Int. Conf. Comput. Sci. Inf. Technol. CSIT 2015*, no. September, pp. 146–153, 2015, doi: 10.1109/stc-csit.2015.7325454.

[10] D. G. Lubas, "Department of defense system of systems reliability challenges," *Proc. - Annu. Reliab. Maintainab. Symp.*, pp. 1–6, 2017, doi: 10.1109/ram.2017.7889676.

[11] L. Fan and Z. Ma, "Tendency analysis of software reliability engineering," *ICRMS'2011 - Saf. First, Reliab. Prim. Proc. 2011 9th Int. Conf. Reliab. Maintainab. Saf.*, pp. 771–773, 2011, doi:10.1109/icrms.2011.5979369.

[12] S. Jayatilleka, "Intersection of systems and reliability engineering during new product development process," *Proc. - Annu. Reliab. Maintainab. Symp.*, vol. 2020-Janua, 2020, doi:10.1109/rams48030.2020.9153653.

[13] F. M. Safie, R. G. Stutts, and Z. Huang, "Reliability and probabilistic risk assessment - How they play together," *Proc. - Annu. Reliab. Maintainab. Symp.*, vol. 2015-May, pp. 1–5, 2015, doi:10.1109/rams.2015.7105058.

[14] J. Luo, H. Li, and S. Wang, "A quantitative reliability assessment and risk quantification method for microgrids considering supply and demand uncertainties Loss of Energy Expected Loss of Load Expected Loss of Power Supply Probability," *Appl. Energy*, vol. 328, no. September, p. 120130, 2022, doi: 10.1016/j.apenergy.2022.120130.

[15] J. Ai, W. Su, and F. Wang, "Software Reliability Evaluation Method Based on a Software Network," *Proc. - 29th IEEE Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2018*, pp. 136–137, 2018, doi:10.1109/issrew.2018.00-15.

[16] A. Boranbayev, S. Boranbayev, and A. Nurusheva, "Development of a software system to ensure the reliability and fault tolerance in information systems based on expert estimates," 2018, doi:10.1007/978-3-030-01057-7_68.

[17] R. Mijumbi, K. Okumoto, A. Asthana, and J. Meekel, "Recent Advances in Software Reliability Assurance," *Proc. - 29th IEEE Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2018*, pp. 77–82, 2018, doi:10.1109/issrew.2018.00-27.

[18] Y. Liu, M. Lu, and B. Xu, "Software reliability case development method based on software reliability characteristic model and measures of defect control," *Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS*, vol. 0, pp. 1–6, 2016, doi:10.1109/icsess.2016.7883004.

[19] Y. Zhao, J. Gong, Y. Hu, Z. Liu, and L. Cai, "Analysis of quality evaluation based on ISO/IEC SQuaRE series standards and its considerations," *Proc. - 16th IEEE/ACIS Int. Conf. Comput. Inf. Sci. ICIS 2017*, pp. 245–250, 2017, doi: 10.1109/icis.2017.7960001.

[20] M. A. Al Imran, S. P. Lee, and M. A. M. Ahsan, "Measuring impact factors to achieve conflict-free set of quality attributes," *2017 IEEE 8th Control Syst. Grad. Res. Colloquium, ICSGRC 2017 - Proc.*, no. August, pp. 174–178, 2017, doi: 10.1109/icsgrc.2017.8070590.

[21] H. Al-Kilidar, K. Cox, and B. Kitchenham, "The use and usefulness of the ISO/IEC 9126 quality standard," *2005 Int. Symp. Empir. Softw. Eng. ISESE 2005*, pp. 126–132, 2005, doi:10.1109/isese.2005.1541821.

[22] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are 'Non-functional' Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice," *Lect. Notes Informatics (LNI), Proc. - Ser. Gesellschaft fur Inform.*, vol. P-267, pp. 105–106, 2016.

[23] S. Zhu, M. Lu, and B. Xu, "Software Reliability Case Development Method Based on the 4+1 Principles," *Proc. - 12th Int. Conf. Reliab. Maint. Safety, ICRMS 2018*, pp. 197–202, 2018, doi:10.1109/icrms.2018.00045.

[24] F. P. Juniawan *et al.*, "E-Voting Software Quality Analysis with McCall's Method," *2020 8th Int. Conf. Cyber IT Serv. Manag. CITSM 2020*, pp. 7–11, 2020, doi: 10.1109/citsm50537.2020.9268854.

[25] J. Zhang, "Field Product Reliability Risk Assessment," *Proc. - Annu. Reliab. Maintainab. Symp.*, vol. 2022-Janua, pp. 1–6, 2022, doi:10.1109/rams51457.2022.9894003.

[26] N. Singh and K. Tyagi, "Important factors for estimating reliability of SOA," *Conf. Proceeding - 2015 Int. Conf. Adv. Comput. Eng. Appl. ICACEA 2015*, pp. 381–386, 2015, doi:10.1109/icacea.2015.7164734.

[27] A. Quyoum, M.-U.-D. Dar, and S. M. K. Quadri, "Improving Software Reliability using Software Engineering Approach- A Review," *Int. J. Comput. Appl.*, vol. 10, no. 5, pp. 41–47, 2010, doi: 10.5120/1474-1990.

[28] B. Kitchenham, "Procedures for Performing Systematic Reviews," *Empir. Softw. Eng.*, vol. 33, no. 2004, pp. 1–26, 2004.

[29] M. Lepmets, E. Ras, and A. Renault, "A quality measurement framework for IT services," *Proc. - 2011 Annu. SRII Glob. Conf. SRII 2011*, pp. 767–774, 2011, doi: 10.1109/srii.2011.84.

[30] S. Yin, Q. Shi, Y. Wang, and C. Chen, "Summary of software reliability Research," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1043, no. 5, 2021, doi: 10.1088/1757-899X/1043/5/052039.

[31] X. Pan and M. Zhang, "Quality and Reliability Improvement Based on the Quality Function Deployment Method," *Proc. - 12th Int. Conf. Reliab. Maint. Safety, ICRMS 2018*, pp. 38–42, 2018, doi:10.1109/icrms.2018.00018.

[32] J. Klohoker, "A risk informed approach to reliability requirements tailoring," *Proc. - Annu. Reliab. Maintainab. Symp.*, vol. 2019-Janua, pp. 1–4, 2019, doi: 10.1109/rams.2019.8768994.

[33] M. Radu, "Reliability and fault tolerance analysis of FPGA platforms," *2014 IEEE Long Isl. Syst. Appl. Technol. Conf. LISAT 2014*, pp. 1–4, 2014, doi: 10.1109/lisat.2014.6845211.

[34] P. Garraghan *et al.*, "Emergent Failures: Rethinking Cloud Reliability at Scale," *IEEE Cloud Comput.*, vol. 5, no. 5, pp. 12–21, 2018, doi:10.1109/mcc.2018.053711662.

[35] J. Wang, "Model of Open Source Software Reliability with Fault Introduction Obeying the Generalized Pareto Distribution," *Arab. J. Sci. Eng.*, vol. 46, no. 4, pp. 3981–4000, 2021, doi: 10.1007/s13369-021-05382-4.

[36] P. S. Sabnis, S. Joshi, and J. Naveenkumar, "A Study on Machine Learning Techniques based Software Reliability Assessment," *4th Int. Conf. Inven. Res. Comput. Appl. ICIRCA 2022 - Proc.*, no. Icirca, pp. 687–692, 2022, doi: 10.1109/icirca54612.2022.9985530.

[37] E. Bagheri and F. Ensan, “Reliability estimation for component-based software product lines,” *Can. J. Electr. Comput. Eng.*, vol. 37, no. 2, pp. 94–112, 2014, doi: 10.1109/cjece.2014.2323958.

[38] Y. Li, W. Wang, and X. Leng, “A Mission Reliability Method (MRM) for Risk Management in the Development of Materiel System,” p. 5, 2010.

[39] E. Cota, “Adjusting reliability predictions for risk,” 2017 Annual Reliability and Maintainability Symposium (RAMS), pp. 1–5, 2017, doi: 10.1109/ram.2017.7889717.

[40] Q. Li, L. Luo, and J. Wang, “Accelerated reliability testing approach for high-reliablity software based on the reinforced operational profile,” *2013 IEEE Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2013*, pp. 337–342, 2013, doi: 10.1109/issrew.2013.6688917.

[41] B. Cukic and J. Dong, “Availability Monitor for a Software Based System,” *Proc. IEEE Int. Symp. High Assur. Syst. Eng.*, pp. 321–328, 2007, doi: 10.1109/hase.2007.49.

[42] L. C. Hao, L. J. Wu, R. Yan, X. Y. Han, and L. L. Tang, “Research on Software Reliability Index Allocation Method Based on Network Architecture,” *Proc. 2019 Int. Conf. Qual. Reliab. Risk, Maintenance, Saf. Eng. QR2MSE 2019*, no. Qr2mse, pp. 551–556, 2019, doi:10.1109/qr2mse46217.2019.9021200.

[43] S. M. Yacoub and H. H. Ammar, “A methodology for architecture-level reliability risk analysis,” *IEEE Trans. Softw. Eng.*, vol. 28, no. 6, pp. 529–547, 2002, doi: 10.1109/tse.2002.1010058.

[44] L. Chang, X. Song, and L. Zhang, “Uncertainty-oriented reliability and risk-based output control for complex systems with compatibility considerations,” *Inf. Sci. (Ny).*, vol. 606, pp. 512–530, 2022, doi:10.1016/j.ins.2022.05.068.

[45] C. Ji, D. Wu, D. Cheng, and Z. Shen, “Software-hardware interdependent reliability assessment technique for software-intensive complex systems,” *ICRMS 2014 - Proc. 2014 10th Int. Conf. Reliab. Maintainab. Saf. More Reliab. Prod. More Secur. Life*, pp. 493–500, 2014, doi: 10.1109/icrms.2014.7107246.

[46] K. Okumoto, A. Asthana, and R. Mijumbi, “BRACE: Cloud-based software reliability assurance,” *Proc. - 2017 IEEE 28th Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2017*, pp. 57–60, 2017, doi:10.1109/issrew.2017.48.

[47] V. Gaur, O. P. Yadav, G. Soni, and A. P. S. Rathore, “A Review of Metrics, Algorithms and Methodologies for Network Reliability,” *IEEE Int. Conf. Ind. Eng. Eng. Manag.*, pp. 1129–1133, 2019, doi:10.1109/ieem44572.2019.8978688.

[48] J. Ludwig, S. Xu, and F. Webber, “Compiling static software metrics for reliability and maintainability from GitHub repositories,” *2017 IEEE Int. Conf. Syst. Man, Cybern. SMC 2017*, vol. 2017-Janua, pp. 5–9, 2017, doi: 10.1109/smc.2017.8122569.