Reliability Risk Assessment Approaches in Software Engineering: A Review Structured by Software Development LifeCycle (SDLC) Phases and Reliable Sub-Characteristics

Lehka Subramanium, Saadah Hassan, Mohd. Hafeez Osman, Hazura Zulzalil
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, 43400, Selangor, Malaysia

***Abstract*—Reliability risk is a critical concern in software development, as failures can result in system downtime, degraded performance, data integrity issues, financial losses and loss of user trust. The increasing complexity of modern systems, driven by dynamic workloads, distributed architecture, and unpredictable interactions, amplifies these risks. In regulated industries like healthcare, finance, and transportation, software reliability directly affects safety, compliance and operational continuity, making robust risk assessment essential. Despite recent development and improvement on numerous reliability risk assessment techniques, system failures continue to be potent, creating concerns on scope, applicability and limitations. This paper will dive deep into evaluating recent methods, the advantages and disadvantages of the application itself, while critically assessing the research gaps. Here, the techniques are categorized across the software development lifecycle (SDLC), to bridge methods to phase-specific reliability needs. Consequently, the paper addresses methodological synthesis of recent practices, identifies segments where existing techniques fail to live up to expectations, and summarize future research directions for achieving more robust and adaptive reliability risk assessment.**

***Keywords*—Reliability; risk assessment; SDLC**

I. Introduction

Reliability is a mandatory quality trait in the software industry, ensuring that systems perform persistently and consistently under specific set of conditions without failure [1]. In the modern age where the world has set foot on AI, achieving the utmost level of reliability is increasingly competitive. Though modern software systems are continuously growing, with each development, the complexity of each system is growing. Besides that, integrating distributed architectures, cloud computing, and the Internet of Things (IoT) further prompt dilemma on reliability concerns. Generally, these systems commonly encounter unprecedent workloads, evolving user demands, hardware failures, and security vulnerabilities, all of which create chaos for operation reliability [2]. Key reliability expectations include handling system failures robustly, ensuring fault tolerance, eliminating data corruption, and mitigating downtime [3]. As software systems grow and expand, the complexity of managing reliability risks becomes a more significant burden. Therefore, identifying leading indicators and mitigating them are crucial. Consequently, risk assessment techniques have become a ground-breaking tool for analyzing and mitigating potential reliability issues throughout the software development lifecycle.

A series of risk assessment techniques [4] have been developed to identify and manage reliability risk. In the past. tools such as Failure Mode and Effects Analysis (FMEA), Fault Tree Analysis (FTA), and Reliability Block Diagrams (RBD) have been significant to identify potential points of failure. In recent years, numerous advanced techniques, including probabilistic risk assessments, machine learning models, and simulation-based techniques, have undoubtedly improved the accuracy of predicting, quantifying reliability risks and efficiency of risk management. These approaches have enabled software engineers to detect weaknesses proactively and develop mitigation strategies before risks manifest into system failures.

Despite these advancements, several challenges remain unsolved. No single technique can comprehensively address all dimensions of reliability risk. Many techniques have limited applicability across diverse system architectures, operational environments, or specific failure modes [5]. Some methods are resource-intensive or not scaled for large, distributed systems, while others lack adaptability for the dynamic nature of modern system. Our previous systematic literature review in Subramanium et al. [6], identified key reliability characteristics, associated risks and evaluation metrics, but did not examine how existing risk assessment techniques map to these characteristics across different phases of the software development lifecycle. This gap highlights the need for a structured synthesis of current techniques to identify the coverage limitations and research opportunities.

Unlike prior reviews that often discuss reliability risk assessment techniques in isolation or without considering their relevance across development phases, this study provides a structured synthesis by mapping techniques to both the software development lifecycle (SDLC) phases and reliability sub-characteristics. This dual-perspective approach clarifies where each technique fits best, what risks it addresses, and where methodological gaps persist despite decades of research [7]. Systematically comparing the advantages, limitations, and domain-specific applications of existing methods, this paper goes beyond descriptive reviews to offer a decision-support view for practitioners selecting suitable techniques at different stages of development. For researchers, the findings highlight

underexplored intersections between risk assessment methods, evolving software architectures, and emerging paradigms like cloud computing, IoT, and AI-driven systems [8] providing a foundation for innovative, adaptive reliability assessment frameworks in future work.

The remainder of this paper is organized as follows: Section II provides the background on reliability key concepts. Section III reviews the related work, including existing literature and applications across different system types. Section IV outlines the methodology used to analyze and compare the identified techniques. Section V presents the results, including the classification of techniques, their advantages, limitations and domain specific applications. Next, Section VI discusses the findings, highlighting research gaps and implications for practice and future research. Finally, Section VII concludes the paper by summarizing the key insights and proposing directions for advancing reliability risk assessment practices

II. Background

A. Software Reliability and its Sub-Attributes

A fundamental quality attribute of a product is the reliability of the product itself. Reliability is defined as its capability to perform functions consistently and persistently under specified conditions for a predetermined period [6],[9]. Reliability has been universally recognized in major software quality models such as McCall, Boehm, FURPS, ISO 9126, and the more recent ISO 25010 [10], [11] all of which singled out reliability as a prominent quality attribute and further classify it into several sub-characteristics. In our most recent work, *A Systematic Literature Review on Characteristics Influencing Software Reliability* (2024), we critically summarized a comprehensive analysis of these sub-characteristics and their function on overall software reliability [6]. Here, software reliability is commonly examined through dimensions such as fault tolerance, recoverability, maturity, and availability. Specifically, fault tolerance emphasize the system's ability to continue operating correctly even in the presence of faults [12], whereas recoverability focuses the capacity to restore optimal operations after a failure with minimum damage [13]. Furthermore, maturity affects the system's stability and defect density over time, indicating the effectiveness of its development and testing processes [13],[14],[15]. Next, the availability dimension focuses on making sure the system remains operational and accessible persistently whenever required. This is a crucial factor in safety-critical domains such as finance and healthcare [13]. Digesting these sub-characteristics provides the fundamental basis for evaluating how recent risk assessment techniques address different factors of software reliability across the software development lifecycle.

B. Risk Assessment in Software Engineering

The flow of risk assessment in software engineering commonly involves identifying potential threats, analyzing their probability, prioritizing them, and executing mitigation steps and monitoring strategies [16]. On the scope of software reliability, risks generally sprout out from factors such as complexity in system, unprecedented workloads, distributed architecture, and operational uncertainties that jeopardizes performance and stability [17]. Lately, various techniques have been deployed to analyze and mitigate such risks, ranging from conventional approaches like Failure Mode and Effects Analysis (FMEA), Fault Tree Analysis (FTA), and Reliability Block Diagrams (RBD) to advanced and dynamic methods such as probabilistic risk assessments, Bayesian networks, and simulation-based techniques [18],[19],[20]. Machine learning [21], [22] and AI-driven models [8],[23] have also been explored to improve predictive accuracy and support real-time decision-making in reliability risk assessment. Despite all these efforts, the gaps exist in understanding their applicability, limitations, and compatibility with different stages of software development lifecycle.

C. Need for SDLC-Based Categorization

The Software Development Lifecycle (SDLC) provides a structured and comprehensive framework for planning, designing, developing, testing, and maintaining software systems [7]. Each segment addresses unique reliability risks and acts as a leading indicator to address and tackle risk that may cost catastrophic failures at latter stage of product development. For instance, poor requirements analysis may blind reliability constraints, while deficient testing could allow risk to bypass undetected until late in deployment. Studies indicate that both the choice of development methodology and the factors stressed at each stage significantly amplify reliability outcomes [7]. In particular, [7] highlights fifteen major factors influencing reliability, with decision made in requirement and design during early stages of development having the most significant impact on the final product's quality. This further reiterates the mandatory need to implement reliability risk assessment techniques across all SDLC, instead of applying them only at later stages of development. By narrowing existing techniques to specific SDLC stages, this paper aims to identify coverage gaps and provide perception into how reliability risk assessment practices can be made more structured, comprehensive, effective, context-aware and aligned with phase specific reliability needs.

III. Related Work

The sky is the limit for reliability risk assessment in software engineering. The field continues to evolve over time to cater to increasing complexity of modern systems and the persistent challenge of mitigating software failures. In the most recent times, studies have revealed a wide range of approaches, ranging from foundational qualitative and quantitative methods to advanced machine learning and artificial intelligence-driven techniques. These developments objectives are to enhance the accuracy, comprehensiveness, and real-time applicability of risk assessment across the Software Development Life Cycle (SDLC).

A recent study in 2023 by Jing et al [24] has investigated on the conventional reliability risk assessment techniques, such as Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA), which have served as primary tools for risk analysis, especially during the initial design phases. For instance, FTA to employ a top-down deductive approach to identify sequences of events leading to a failure, while FMEA uses a bottom-up inductive method to systematically analyse

potential failure modes and their consequences. Event Tree Analysis (ETA) is often applied alongside these methods to explore the range of possible outcomes following an initiating event. Despite the widespread use for equipment failure diagnosis, root cause analysis, and preliminary safety assessments, these methods face several challenges. A paper in 2022 by Shayan et al [25] evaluated the conventional FMEA, though comprehensive and computationally efficient, struggles with data fragmentation, inconsistent formats, and limitations in the Risk Priority Number (RPN) calculation, which often gives equal weight to severity, occurrence, and detection, sometimes leading to mis prioritized risks. Similarly, FTA encounters difficulties in handling dynamic system characteristics and large-scale data, and all three methods remain largely manual and prone to subjectivity, especially when systems undergo frequent updates.

To further mitigate uncertainty and data incompleteness in traditional methods, researchers have incorporated Fuzzy Logic into risk assessment frameworks, giving rise to Fuzzy Fault Trees (FFT), Fuzzy Event Trees (FET), and Fuzzy FMEA (F-FMEA) [18]. Although conventional techniques focus on precise probabilities, fuzzy methods which enables the use of linguistic variables to shortlist likelihoods and outcomes, making these methods suitable for data-scarce environments. For example, F-FMEA improves conventional RPN calculations by singularly analysing significance, tendency of occurrence, and detection, reducing human impact and enhancing sensitivity to minute parameter alteration. Consequently, integration of fuzzy reasoning has paved ways to a more suitable and adaptive risk prioritization in complex and uncertain environments. A similar approach using the fuzzy FMEA has been applied in the field of cyber security [26] and IT-governance [27] to evaluate risk maturity levels. Building on these successful enhancement, significant research effort continues to improve integration of conventional methods with advanced modelling, automation and analytics techniques for more comprehensive reliability risk assessment [4],[28],[29].

Extending this effort, recent research has focused on addressing randomness and dynamic behavior in systems. For instance, probabilistic techniques such as Markov Chain Models (MCM) and Monte Carlo Simulation (MCS) have been introduced [30],[31],[32],[33]. As for MCM, [30] and [31] have provided a framework for modeling stochastic transitions between system states, while [33] and [34] enables probabilistic exploration of risk scenarios under uncertainty in MCS. However, a specific study on embedded system by [32] discussed that MCS becomes computationally expensive if system complexity and the number of variables increase, limiting its applicability to large-scale systems.

Recently, one other significant advancement of Artificial Intelligence (AI) has opened new opportunities for reliability risk assessment. AI leverage machine learning, deep learning and knowledge representation methods to automate fault detection, predicting failure probabilities and analyse large scale data in real time. With this, new risk assessment techniques were explored by a recent study in 2024 [8], Knowledge Graphs (KGs), in particular, the Fault Knowledge Graphs (FKGs), vector machines and neural network [23]. These AI powered techniques enable adaptive and self-learning reliability risk assessment frameworks. The author has focused on using FKGs to assess key parameters like reliability of the returned information. Similarly, in 2025 [23], Zhao et all has evaluated the use of FKGs to design a multi-level modular structure on an electric power system using neural network and support vector machines by comparing against conventional methods. With all the efforts, a paper in 2023 [35] has critically provided analysis of FKGs that it faces insufficient standardized modelling practices and complexity in extracting information from unstructured data sources. Nevertheless, by further enhancing studies and research on automated techniques and ontology construction, the true potential of KGs in reliability engineering can be realised and it will be groundbreaking.

In a nutshell, the evolution of reliability risk assessment in software engineering shows a shift in pathway from a static framework, toward more dynamic, automated, intelligent, and data-driven techniques. Conventional methods like FTA and FMEA do bring a valuable foundation but tend to get ineffective with dynamic system behaviour and bulk datasets [24]. Increasingly, probabilistic approaches, fuzzy logic, and knowledge graphs address a small sample limitation by introducing probabilistic reasoning, uncertainty modeling, and structured data integration.

Future research should address challenges such as knowledge graph standardization, hybridizing AI and probabilistic methods for improved explainability, and developing computationally efficient techniques for large-scale, real-time applications. Such advancements will be critical for ensuring the reliability and resilience of next-generation software systems

IV. Methodology

This study adopts a structured review approach to analyze software reliability risk assessment techniques across the software development lifecycle (SDLC). The primary objective is to explore how various techniques have been applied, their strengths and limitations, and their suitability for addressing reliability risks in modern software environments.

As illustrated in Fig 1, the review process follows five interconnected stages:

- Step 1: Literature Review and Data Collection
- Step 2: Selection of Relevant Techniques
- Step 3: Mapping Techniques to SDLC Phases
- Step 4: Comparative Analysis of Techniques
- Step 5: Data Synthesis and Research Gaps

Each stage is explained in the following subsections, ensuring a clear and logical flow from gathering information to drawing insights and identifying research gaps.

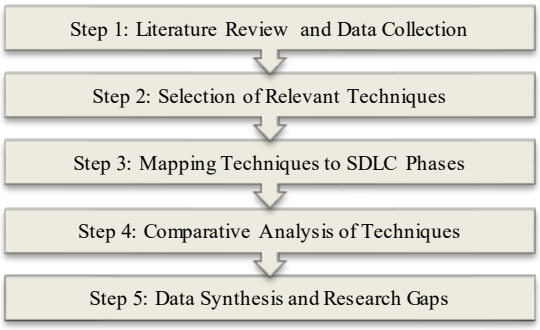

Fig. 1. Design procedure.

A. Literature Review and Data Collection

The first stage involved peer-reviewed studies from academic databases, including IEEE Xplore, Scopus, ACM Digital Library, and Google Scholar. Keywords such as "software reliability," "risk assessment techniques," "reliability sub-characteristics," and "reliability risk analysis" were combined using Boolean operators to ensure comprehensive coverage. Also, terms such as "reliability AND risk matrix", "reliability AND Probabilistic Risk Assessment" and with other risk assessment techniques uniquely were employed to retrieve studies emphasizing reliability-related risk modelling methods. This approach minimized irrelevant results and ensured that only techniques applicable to software reliability evaluation were captured.

To maintain relevance, only studies published in English between 2015 and 2025 were included. Non-peer-reviewed sources, duplicate studies, and research lacking methodological detail were excluded.

B. Selection of Relevant Techniques

The next step was to select the techniques that were most frequently discussed and demonstrated clear relevance to software reliability risk assessment. A simple rating approach was used to give a more objective selection basis.

Each technique was rated on aspects such as:

- Frequency of application in studies
- Coverage of reliability sub-characteristics (referring to the standard quality model, ISO 25010)
- Clarity and completeness of methodological description
- Reported advantages and limitations

This allowed the review to prioritize techniques with strong academic support while still discussing fewer common methods for completeness.

C. Mapping Techniques to SDLC Phases

Furthermore, techniques were classified according to the phases of the Software Development Life Cycle (SDLC) to provide practitioners with a clear reference on which techniques are most suitable for specific stages of software development, bridging the gap between research insights and real-world application.

D. Comparative Analysis of Techniques

Once the techniques were aligned to SDLC phases, a comparative analysis was undertaken to present the information in a structured and transparent manner. Specifically, each technique was analyzed across the following dimensions:

- Risk assessment techniques: Name of the technique
- Overview: Brief description of its approach or methodology
- Domain application: Software domains or system types where it has been applied (example: IoT, cloud computing, real-time systems, AI)
- Coverage of reliability sub-characteristics: Mapping to reliability characteristics based on software quality model ISO 25010.
- Advantages: Reported strengths such as accuracy, scalability, cost-effectiveness.
- Limitations- Key challenges like implementation complexity or lack of empirical validation.
- References: Key studies supporting the analysis

This classification ensures a comprehensive, phase-oriented perspective for evaluating the suitability of risk assessment techniques in modern software environments, enabling both researchers and practitioners to select methods best aligned with their system requirements.

E. Data Synthesis and Research Gaps

After the comparative analysis, findings were synthesized to identify patterns, trends, and gaps. This stage helped highlight gaps in the literature, such as the lack of empirical validation for certain techniques or limited application in specific software domains, providing directions for future studies.

Despite its structured approach, this review is limited by potential publication bias due to reliance on peer-reviewed studies in English and selected academic databases. Furthermore, the analysis depends on secondary data without empirical testing, which may restrict the generalizability of conclusions. Acknowledging these limitations provides transparency and highlights opportunities for empirical validation in future studies.

V. Results

Table I presents a phase-wise summary of software reliability risk assessment techniques across the analysis, development, and testing stages of the software development, and testing stages of the software development lifecycle (SDLC). Categorizing techniques by SDLC phase ensures that the distinct risk management requirements of each stage are adequately addressed, enabling practitioners to apply the most suitable methods at the right time. Building on this table, the subsequent discussion evaluates each technique in terms of overview, domain application, coverage of reliability sub characteristics, providing a comprehensive understanding of their strengths, weaknesses for modern software environments.

TABLE I SUMMARY OF SOFTWARE RELIABILITY RISK ASSESSMENT TECHNIQUES ACROSS SDLC PHASES

Risk Assessment Techniques	Overview	Domain Application	Coverage of reliability sub-attributes	Advantages	Limitations	References
REQUIREMENTS						
Risk Matrix	A qualitative or semi-quantitative tool that categorizes risks by severity and likelihood to prioritize them.	- Safety-Critical Systems - Industrial Automation & Manufacturing - Cloud Computing - Real-time and Embedded systems - Software Engineering	Availability, Maturity, Recoverability	- A transparent representation of risk levels using likelihood and consequence scales - Low implementation cost - Support quick decision making - Customizable (project size, different domains, risk categories) - Communication friendly	- Less efficient in modeling complex reliability behaviors. - Subjectivity in risk scoring - Lack of probabilistic measures - Limited predictive capability for dynamic or real-time environments where risk levels evolve quickly	[36], [37], [38]
Risk-Driven Requirement Engineering	A systematic approach to derive and prioritize requirements based on identified risks.	- Safety-Critical (aerospace, healthcare) - Large-scale enterprise software telecommunication system) -Agile and Incremental Development	Maturity, Fault Tolerance:	- Early risk identification - Improve requirement quality - Support requirement prioritization - Reduce rework costs.	- Dependent on expert judgement - Time-consuming for large projects - Limited tool integration - Subjective risk estimation	[20], [39]
Failure-Oriented Requirements Engineering (FORE)	Focus on identifying failure modes and their impact on requirements to mitigate risks early.	- Safety-critical - Large scale industrial systems - Real-time systems - IoT applications	Maturity Fault Tolerance	- Failure identification at the requirement phase - Support early fault analysis - Integrates with safety standards - Support systematic traceability from failures to requirements	- Require detailed system knowledge early in the system development - Time intensive for complex systems with many failures modes - Subjectivity on expert driven failure modelling	[40]
Goal-Oriented Risk Assessment (GORA)	Identifies risks through the lens of system goals, ensuring alignment between objectives and risk mitigation.	- Complex socio-technical system - Business critical software - Safety systems - Requirement engineering - Decision support systems - Policy driven IT systems	Maturity	- Improve traceability from risks to system objectives - Useful in multi-stakeholder environments - Support what-if analysis - Provides structured modeling linking risks to goals, obstacles and mitigations.	- Limited reliability quantitative evaluation - High effort for large systems with many goals - Require tools for efficient goal modeling.	[41], [42]

Risk-Based Requirement Prioritization	Prioritize requirements based on their associated risks to ensure critical risks are addressed first.	- Agile projects, incremental development, time constrained software releases -Cloud software, e-commerce platforms, enterprise applications	Maturity Fault Tolerance	- Ensure critical, high-risk requirements are developed first - Cost-effective - Fits well with Agile and iterative methodologies like Scrum - Support risk-driven release planning	- May ignore low-risk requirements that could be important for reliability - Subjective prioritization - Limited support for quantitative risk modelling	[43], [33]
Probabilistic Risk Assessment (PRA) in Requirements	Quantifies the likelihood of risks in meeting requirements by analyzing statistical probabilities	- Nuclear system - Defense system - Autonomous vehicles - Mission critical software - Model-based systems engineering for safety and reliability certifications	Maturity Availability Fault Tolerance	- Provide quantitative, probabilistic measures of risk. - Support early decision making under uncertainty using likelihood and consequences modelling - integrates with Fault Tree Analysis (FTA) and Event Tree Analysis (ETA) - Enables reliability-based requirement verification early	- Require extensive data for accurate probability estimation - High modelling complexity and computational costs for large systems - Steep learning curve	[44], [33], [45], [46]
DESIGN						
Hazard and Operability Study (HAZOP)	A structured and systematic technique to identify hazards and operability issues in processes or systems by evaluating deviations from design intent.	- Process control system - Chemical plants - Industrial automation - Safety critical software	Fault Tolerance, Availability	- Structured approach ensures no risk scenario is overlooked. - Suitable for early phases, enabling proactive risk mitigation. - Adopted in safety standards	- Require time-consuming workshops for comprehensive coverage. - Largely qualitative - Subjective to expert knowledge accuracy.	[47], [48],
Fault Tree Analysis (FTA)	A top-down, deductive approach to identify causes of system-level failures by analyzing fault logic.	- Aerospace, nuclear power plants - Automotive safety systems.	Availability, Maturity	- Easy-to-understand fault visualization. - Supports probabilistic failure estimation using failure rate data. - Standardized technique.	- Assumes independence of failures, unrealistic for complex systems. - Static analysis, cannot handle event sequences well. - Large systems produce complex, unwieldy trees.	[18], [35], [29]
Event Tree Analysis (ETA)	A forward-looking technique that models the possible outcomes of an initiating event through branching event sequences.	- Nuclear safety - Aviation risk modeling - Chemical plants.	Availability:	- Forward looking approach complements FTA - Support probability mapping	- Event paths multiply quickly in complex systems - Quality depends on correct events - Assumption on event independence	[18], [24]
Reliability Block Diagram (RBD)	Graphical representation of system reliability structure by connecting components in series or	- Aerospace - Manufacturing - Power Systems	Availability, Fault Tolerance	- Provide graphical modeling	- Assumes static configuration	[15]

	parallel to depict dependency			- Suitable to identify critical components	- Limited for dynamic or stochastic systems	
Monte Carlo Simulation	Uses random sampling and statistical modeling to simulate the probability of different outcomes and assess risk.	- Finance - Cloud Systems - Telecommunication	Fault Tolerance, Availability, Maturity	- Captures uncertainty and probabilistic behavior - Handles complex, interdependent systems	- Requires high computational cost - Accuracy depends on number of simulations	[33], [34]
Markov Analysis and Modelling	Analyzes system states and transitions using probabilistic modeling for systems with random state changes over time.	- Embedded Systems, - Network Systems, - Health Systems	Availability, Recoverability, Fault Tolerance	- Models stochastic and time-dependent failures - Supports repairable system modeling	- Assumes exponential failure distributions	[44], [46] [30], [32], [31],
Petri Nets	A graphical and mathematical modeling tool to represent concurrent, distributed, and dynamic system processes.	- Industrial Control Systems - Distributed Systems	Fault Tolerance, Recoverability	- Models stochastic and time-dependent failures -Supports repairable system modeling	- Modeling complexity increases with system size	[4], [49]
Stochastic Petri Nets (SPN)	Extends Petri Nets with stochastic timing to model system behaviors and performance under uncertainty.	- Real-time Systems - IoT, - Cyber-Physical Systems	Fault Tolerance, Availability, Recoverability	- Captures stochastic behavior and timing aspects - Supports dynamic analysis	- Higher computational cost; requires specialized expertise	[50]
Bayesian Networks	A probabilistic graphical model that represents variables and their dependencies using directed acyclic graphs.	- Autonomous Vehicles - Smart Grids, Healthcare	Fault Tolerance, Availability, Recoverability, Maturity	- Probabilistic reasoning under uncertainty -Supports learning from data and evidence updating	- Scalability issues for large systems. - Data dependency specification challenges	[4],[51]
Fuzzy Logic	Mathematical approach for handling uncertainty and imprecision in risk assessment.	- IoT - Edge Computing - Decision-Support Systems	Fault Tolerance, Recoverability, Maturity	- Handles imprecise, linguistic, or uncertain data well - Suitable for expert-judgment-based analysis	- Subjectivity in membership function design - Lacks standardization across domains -	[18]
DEVELOPMENT						
Failure Modes and Effects Analysis (FMEA)	A systematic approach to identify potential failure modes, their causes, and effects to prioritize corrective actions.	Fault Tolerance, Availability, Recoverability	- Automotive - Aerospace - Industrial Systems	- Structured and systematic - Early defect identification	- Time-consuming - qualitative in nature - Subjective to expert knowledge	[24], [29]
Code Review and Static Code Analysis	Evaluates code quality by manually reviewing or using automated tools to detect bugs, vulnerabilities, and design flaws.	Maturity, Fault Tolerance	- Software Development - Web Applications like Banking Systems	- Early defect detection before runtime - Automated tools available - Cost-effective for initial phases	- Limited to static properties - Cannot detect runtime or dynamic behavior issues	[52]
Reliability Growth Models	Models that predict system reliability improvement over time as defects are identified and corrected during testing.	Maturity, Availability	- Telecommunication Systems	- Focus reliability improvement over testing cycles - Quantitative assessment of defect detection	- Assumes failure patterns follow statistical distributions	[53], [54], [55]
Dynamic Analysis and	Examine system behavior during execution to identify	Fault Tolerance, Availability, Recoverability	- Cloud Systems - IoT	- Provides real-time insights	- Performance overhead	[56]

Runtime Monitoring	potential runtime errors, resource bottlenecks, or reliability risks.		- Autonomous Systems	- Detects runtime anomalies and adaptive risks	- May be complex to implement at scale	
Test Case Prioritization and Reliability Testing	Prioritizes test cases based on their likelihood of revealing faults and evaluates reliability through systematic testing.	Fault Tolerance, Availability, Maturity	- Embedded Systems, Telecom	- Ensures critical functionalities tested first - Improves fault detection efficiency	- Effectiveness depends on quality of test cases - May require domain-specific customization	[43], [57]
Model-Based Reliability Assessment	Uses formal models to simulate and evaluate system reliability based on its design, configuration, and usage scenarios.	Fault Tolerance, Availability, Recoverability, Maturity	- Safety-Critical Systems (Aerospace, Healthcare)	- Formal and structured modeling - Supports early-stage risk evaluation - Allows simulation of different scenarios	- High modeling effort - Requires specialized skills and accurate system specifications	[33], [50], [58]
TESTING						
Reliability Growth Testing using the Software Reliability Growth Model (SRGM) technique	Evaluates the improvement in system reliability by statistical models that estimate the reliability of software systems by analyzing failure data over time to predict future reliability.	Maturity, Fault Tolerance	- Software Development - Telecommunications - Defense	- Quantitative evaluation of reliability improvement over time	- Requires historical failure data - Ineffective to dynamic changing environments	[53], [59]
Fault Injection Testing	Deliberately introduces faults into a system to evaluate their fault tolerance and recovery mechanisms.	Fault Tolerance, Recoverability	- Cloud Systems - Automotive - Safety-Critical Software	- Evaluates system robustness under failure conditions - Reveals hidden vulnerabilities	- May be costly or risky in production-like environments	[60]
Model-Based Reliability Testing (MBT)	Uses formal models of system behavior to derive test cases and evaluate reliability.	Fault Tolerance, Availability, Maturity	- Aerospace, - Healthcare - Embedded Systems	- Systematic test generation from formal models - Improves coverage and defect detection	- High modeling effort - Requires formal specifications	[61], [62]
Reliability prediction models	Uses mathematical models to estimate system reliability based on design parameters and historical data.	Fault Tolerance, Availability, Maturity	- Defense - Aerospace - Software Engineering	- Supports early estimation of reliability before deployment - Quantitative risk assessment	- Accuracy depends heavily on data quality and assumptions	[22]
Scenario-based testing for reliability	Focuses on testing specific scenarios to evaluate reliability under predefined conditions.	Availability, Fault Tolerance, Recoverability	- Automotive - IoT - Real-Time Systems	- Captures real-world operational conditions - Effective for stress and edge-case testing	- Scenario design complexity - May not cover unexpected runtime behaviors	[63]
Fault Knowledge Graphs	Knowledge-driven approach where information about system faults, failure modes, causes, and their interrelationships is represented as a graph structure	Availability, Fault Tolerance, Recoverability	- AI Systems - Cloud Computing - Complex Software Architectures	- Enables fault pattern analysis using structured knowledge representation - Supports root cause analysis	- Still emerging - Limited standardization; requires high-quality knowledge bases	[35]

VI. Discussion

The categorization of reliability risk assessment techniques across software development lifecycle (SDLC) phases, as summarized in Table I, provides a structured perspective on when and how these methods are applied by linking techniques to specific phases, analysis, design, development, and testing. It also enables practitioners to select methods that align with the unique risk management needs of each stage, avoiding the common pitfall of relying on a one-size-fits-all approach.

In the requirement phase, techniques such as Risk Matrix, Risk-Driven Requirement Engineering, Failure-Oriented Requirements Engineering (FORE), Goal-Oriented Risk Assessment (GORA), Risk-Based Requirement Prioritization, and Probabilistic Risk Assessment (PRA) enable early-stage identification and prioritization of risks.

Next, in the design phase, techniques such as Hazard and Operability Study (HAZOP), Fault Tree Analysis (FTA) , Event Tree Analysis (ETA), Reliability Block Diagrams (RBD) [15], Monte Carlo Simulation, Markov Models, Petri Nets [35], Stochastic Petri Nets (SPN), Bayesian Networks [64], and Fuzzy Logic provide deeper insights into system behavior and potential failure paths.

Further, the development phase introduces Failure Mode and Effects Analysis (FMEA), Code Review and Static Code Analysis, Reliability Growth Models [54], Dynamic Analysis and Runtime Monitoring, Test Case Prioritization and Reliability Testing, and Model-Based Assessment to ensure software reliability as code implementation progresses [4].

Lastly, in the testing phase, as systems transition into validation, techniques like Software Reliability Growth Models (SRGM) [59], Fault Injection, Model-Based Reliability Testing, Reliability Prediction Models, Scenario-Based Testing for Reliability, and Fault Knowledge Gaps Analysis ensure comprehensive risk evaluation under realistic operational conditions.

Despite the availability of diverse risk assessment techniques across the SDLC, one major research gap [65] lies in the fragmented treatment of reliability sub-characteristics [64]. Many techniques either focus on overall system reliability without distinguishing between sub-characteristics such as fault tolerance, availability, recoverability, and maturity, or they emphasize only one aspect while neglecting others [4]. For example, Fault Tree Analysis (FTA) and Reliability Block Diagrams (RBD) primarily evaluate system failure probabilities and availability but provide little insight into recoverability or fault tolerance mechanisms. Conversely, Bayesian Networks and Stochastic Petri Nets (SPN) are strong in modeling fault tolerance and dynamic behavior but are rarely extended to evaluate maturity or maintainability, leading to incomplete risk profiles. This fragmented coverage hinders practitioners from understanding how different reliability sub-attributes interact, for instance, a system might have high fault tolerance yet fail to recover quickly from unexpected outages, compromising operational continuity [66].

Another critical gap concerns the domain-specific nature of reliability requirements. Many existing techniques were designed for general-purpose reliability analysis and do not adequately address domain-driven reliability priorities [67]. For example, recoverability is crucial in financial systems where transaction consistency and disaster recovery are essential, whereas fault tolerance and real-time failure detection dominate autonomous vehicle and industrial automation domains. Similarly, availability and service continuity are top priorities in telecommunication networks and cloud platforms, where downtime directly impacts service quality and revenue. However, most current methods fail to provide customized reliability assessment frameworks that align with the unique risk profiles, operational constraints, and regulatory requirements of these domains [68].

Moreover, as emerging domains such as AI-driven applications, IoT ecosystems, and cyber-physical systems introduce dynamic operational behaviors, data uncertainty, and real-time decision-making, conventional methods like Reliability Growth Models struggle to capture the evolving risk landscape. This highlights [12] the need for adaptive, domain-aware risk assessment approaches that integrate multiple sub-characteristics, model interdependencies, and align reliability priorities with the specific operational and safety requirements of diverse application areas.

VII. Conclusion and Future Work

The study highlighted the evolution of reliability risk assessment techniques across software lifecycle phases, revealing significant progress in modeling, analysis, and prediction. However, findings indicate that despite methodological diversity, many approaches remain phase-specific, static, or computationally demanding, limiting their adaptability to dynamic, data-rich software environments. Moreover, the absence of standardized evaluation benchmarks, limited domain-specific tailoring, and poor alignment with modern development practices such as agile and DevOps hinder practical adoption.

Future research should focus on developing integrated, adaptive, and benchmarked reliability frameworks that balance accuracy, scalability, and usability while accommodating the complexity and dynamism of modern software systems. Emphasis on real-time analytics, AI-driven risk prediction, automated scenario simulation, and cross-domain customization will be essential. Furthermore, extending research toward edge computing, cyber-physical systems, and autonomous platforms offers opportunities to validate and refine techniques under safety-critical and resource-constrained conditions, ensuring reliability assurance remains robust in next-generation software landscapes.

Acknowledgment

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

References

[1] 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.

[2] Y. Zhao, P. Li, J. Deng, M. Gao, Z. Wang, and X. Fan, “Reliability Evaluation Method of Distribution Network Considering Differential Reliability Requirements of End Users,” Proc. 2021 IEEE 4th Int. Electr. Energy Conf. CIEEC 2021, pp. 3–8, 2021, doi: 10.1109/CIEEC50170.2021.9510553.

[3] P. Rotella and S. Chulani, “SRC ratio method: Benchmarking software reliability,” Proc. - 2017 IEEE 28th Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2017, pp. 61–64, 2017, doi: 10.1109/ISSREW.2017.75.

[4] S. Kabir and Y. Papadopoulos, “Applications of Bayesian networks and Petri nets in safety, reliability, and risk assessments: A review,” Saf. Sci., vol. 115, no. November 2018, pp. 154–175, 2019, doi: 10.1016/j.ssci.2019.02.009.

[5] 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.

[6] L. Subramanium, S. Hassan, M. H. Osman, and H. Zulzalil, "A Systematic Literature Review on Characteristics Influencing Software Reliability," Int. J. Informatics Vis., vol. 8, no. 4, pp. 2344–2353, 2024, doi: 10.62527/joiv.8.4.3665.

[7] V. Yakovyna, M. Seniv, and I. Symets, "The Relation between Software Development Methodologies and Factors Affecting Software Reliability," Int. Sci. Tech. Conf. Comput. Sci. Inf. Technol., vol. 1, pp. 377–381, 2020, doi: 10.1109/CSIT49958.2020.9321937.

[8] G. Spasova and D. Dinev, "Exploring the Effectiveness and Reliability of Artificial Intelligence," CIEES 2024 - IEEE Int. Conf. Commun. Information, Electron. Energy Syst., pp. 1–5, 2024, doi: 10.1109/CIEES62939.2024.10811341.

[9] 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.

[10] 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.

[11] P. Nistala, K. V. Nori, and R. Reddy, "Software quality models: A systematic mapping study," Proc. - 2019 IEEE/ACM Int. Conf. Softw. Syst. Process. ICSSP 2019, pp. 125–134, 2019, doi: 10.1109/ICSSP.2019.00025.

[12] K. E. De Souza and F. C. Ferrari, "A Systematic Review of Fault Tolerance Techniques for Adaptive and Context-Aware Systems," Proc. - 2022 IEEE Int. Conf. Auton. Comput. Self-Organizing Syst. ACSOS 2022, pp. 21–30, 2022, doi: 10.1109/ACSOS55765.2022.00020.

[13] E. Ismail, N. Utelieva, A. Balmaganbetova, and S. Tursynbayeva, "The choice of measures reliability of the software for space applications," 2nd Int. Conf. Electr. Commun. Comput. Eng. ICECCE 2020, no. June, pp. 12–13, 2020, doi: 10.1109/ICECCE49384.2020.9179411.

[14] X. H. Wang, Y. X. Li, W. H. Fan, J. Q. Xuan, L. Z. Wang, and M. M. Mu, "Evaluation of product maturity based on quality characteristic," IEEE Int. Conf. Ind. Eng. Eng. Manag., vol. 2016-January, pp. 742–746, 2016, doi: 10.1109/IEEM.2015.7385746.

[15] J. Lucas, A. Thiraviam, A. Elshennawy, and A. M. Albar, "The effectiveness of reliability programs and tools based on design maturity and complexity," Proc. - Annu. Reliab. Maintainab. Symp., pp. 1–5, 2017, doi: 10.1109/RAM.2017.7889658.

[16] S. D. S. Lopes, I. G. Vargas, A. L. De Oliveira, and R. T. V. Braga, "Risk Management for System of Systems: A Systematic Mapping Study," Proc. - 2020 IEEE Int. Conf. Softw. Archit. Companion, ICSA-C 2020, pp. 258–265, 2020, doi: 10.1109/ICSA-C50368.2020.00050.

[17] 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.

[18] C. Quan, L. Lingqiang, Y. Dongping, and Y. Gu, "A reliability risk analysis method based on the fuzzy fault tree and fuzzy event tree," Proc. 2016 11th Int. Conf. Reliab. Maintainab. Saf. Integr. Big Data, Improv. Reliab. Serv. Pers. ICRMS 2016, 2017, doi: 10.1109/ICRMS.2016.8050082.

[19] 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.

[20] O. T. Arogundade, S. Misra, O. O. Abayomi-Alli, and L. Fernandez-Sanz, "Enhancing Misuse Cases with Risk Assessment for Safety Requirements," IEEE Access, vol. 8, pp. 12001–12014, 2020, doi: 10.1109/ACCESS.2019.2963673.

[21] M. Banga, A. Bansal, and A. Singh, "Implementation of Machine Learning Techniques in Software Reliability: A framework," 2019 Int. Conf. Autom. Comput. Technol. Manag. ICACTM 2019, pp. 241–245, 2019, doi: 10.1109/ICACTM.2019.8776830.

[22] C. Ji, "Reliability Evaluation and Prediction of Mechanical System Based on Machine Learning Technology," IEEE 1st Int. Conf. Ambient Intell. Knowl. Informatics Ind. Electron. AIKIIE 2023, pp. 1–5, 2023, doi: 10.1109/AIKIIE60097.2023.10390487.

[23] Z. Zhang and X. Wang, "Design and Research of an Electric Power System Reliability Evaluation Model Based on Artificial Intelligence Algorithms," 2025 5th Asia-Pacific Conf. Commun. Technol. Comput. Sci., pp. 149–155, 2025, doi: 10.1109/ACCTCS66275.2025.00034.

[24] X. Hu, J. Liu, H. Dou, H. Chen, and Y. Zhang, "Automatic Generation of Component Fault Trees from AADL Models for Design Failure Modes and Effects Analysis," IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS, pp. 550–561, 2023, doi: 10.1109/QRS60937.2023.00060.

[25] S. Kumar Akula, H. Salehfar, and S. Behzadirafi, "Comparision of Traditional and Fuzzy Failure Mode and Effects Analysis for Smart Grid Electrical Distribution Systems," 2022 North Am. Power Symp. NAPS 2022, pp. 1–6, 2022, doi: 10.1109/NAPS56150.2022.10012165.

[26] N. A. Chandra, A. A. Putri Ratna, and K. Ramli, "Development of a cyber-situational awareness model of risk maturity using fuzzy fmea," 2020 Int. Work. Big Data Inf. Secur. IWBIS 2020, pp. 127–136, 2020, doi: 10.1109/IWBIS50925.2020.9255543.

[27] U. Yudatama and R. Sarno, "Evaluation maturity index and risk management for it governance using Fuzzy AHP and Fuzzy TOPSIS (case Study Bank XYZ)," 2015 Int. Semin. Intell. Technol. Its Appl. ISITIA 2015 - Proceeding, pp. 323–327, 2015, doi: 10.1109/ISITIA.2015.7220000.

[28] J. Cooper, "Implementation of the Product Reliability Program," 2024 Pan Pacific Strateg. Electron. Symp. Pan Pacific 2024, pp. 1–6, 2024, doi: 10.23919/PanPacific60013.2024.10436446.

[29] P. Fithri, N. A. Riva, L. Susanti, and B. Yuliandra, "Safety analysis at weaving department of PT. X Bogor using Failure Mode and Effect Analysis (FMEA) and Fault Tree Analysis (FTA)," 2018 5th Int. Conf. Ind. Eng. Appl. ICIEA 2018, pp. 382–385, 2018, doi: 10.1109/IEA.2018.8387129.

[30] Y. Jun, J. Chenyu, X. Zhihui, L. Mengkun, and Y. Ming, "Markov/CCMT_ Towards an integrated platform for dynamic reliability and risk analysis," Process Saf. Environ. Prot., vol. 155, pp. 498–517, 2021, doi: 10.1016/j.psep.2021.09.043.

[31] M. F. Aly, I. H. Afefy, R. K. Abdel-Magied, and E. K. A. Elhalim, "A comprehensive model of reliability, availability, and maintainability (RAM) for industrial systems evaluations," Jordan J. Mech. Ind. Eng., vol. 12, no. 1, pp. 59–67, 2018.

[32] S. Shu, Y. Wang, and Y. Wang, "An approach to architecture-based fault tolerance evaluation with fault propagation," Proc. 2015 1st Int. Conf. Reliab. Syst. Eng. ICRSE 2015, vol. 56, pp. 3–9, 2015, doi: 10.1109/ICRSE.2015.7366478.

[33] Y. Jie, W. Wang, G. Wang, and M. Zhang, "Reliability assessment of high microgravity science experiment system based on probabilistic risk assessment," Proc. 2016 Progn. Syst. Heal. Manag. Conf. PHM-Chengdu 2016, pp. 1–6, 2017, doi: 10.1109/PHM.2016.7819797.

[34] M. I. Lunesu, R. Tonelli, L. Marchesi, and M. Marchesi, "Assessing the risk of software development in agile methodologies using simulation," IEEE Access, vol. 9, pp. 134240–134258, 2021, doi: 10.1109/ACCESS.2021.3115941.

[35] L. Shen, H. Tang, L. Wang, J. Cai, and X. Cui, "A Fault Knowledge Graph Creation Method and Application based on Fault Tree Analysis and Failure Mode, Effects and Criticality Analysis," Proc. 2023 IEEE 3rd Int. Conf. Inf. Technol. Big Data Artif. Intell. ICIBA 2023, vol. 3, no. Iciba, pp. 35–39, 2023, doi: 10.1109/ICIBA56860.2023.10165591.

[36] M. Blumenschein, J. Spasic, J. Steckert, and J. Uythoven, "An Approach to Reliability Assessment of Complex Systems at CERN," Proc. - Annu. Reliab. Maintainab. Symp., vol. 2019-Janua, pp. 1–6, 2019, doi: 10.1109/RAMS.2019.8769004.

[37] P. Souvannalath, S. Premrudeepreechacharn, and K. Ngamsanroaj, "Determining Power Transformer Maintenance Plan Using Three-Dimensional Risk Matrix: Note: Sub-titles are not captured in Xplore and should not be used," 2022 IEEE Int. Conf. Power Syst. Technol. Embrac. Adv. Technol. Power Energy Syst. Sustain. Dev. POWERCON 2022, pp. 1–6, 2022, doi: 10.1109/POWERCON53406.2022.9929632.

[38] Z. Danlin, J. Han, S. Jialiang, and Y. Lin, "A risk assessment approach based on fuzzy 3D risk matrix for network device," 2016 2nd IEEE Int.

Conf. Comput. Commun. ICCC 2016 - Proc., pp. 1106–1110, 2017, doi: 10.1109/CompComm.2016.7924876.

[39] F. Wiesweg, A. Vogelsang, and D. Mendez, "Data-driven Risk Management for Requirements Engineering: An Automated Approach based on Bayesian Networks," Proc. IEEE Int. Conf. Requir. Eng., vol. 2020-Augus, pp. 125–135, 2020, doi: 10.1109/RE48521.2020.00024.

[40] 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.

[41] F. Başak Aydemir, P. Giorgini, and J. Mylopoulos, "Multi-objective risk analysis with goal models," Proc. - Int. Conf. Res. Challenges Inf. Sci., vol. 2016-Augus, 2016, doi: 10.1109/RCIS.2016.7549302.

[42] D. Alrajeh, A. Van Lamsweerde, J. Kramer, A. Russo, and S. Uchitel, "Risk-driven revision of requirements models," Proc. - Int. Conf. Softw. Eng., vol. 14-22-May-, pp. 855–865, 2016, doi: 10.1145/2884781.2884838.

[43] C. Hettiarachchi and H. Do, "A Systematic Requirements and Risks-Based Test Case Prioritization Using a Fuzzy Expert System," Proc. - 19th IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2019, pp. 374–385, 2019, doi: 10.1109/QRS.2019.00054.

[44] R. G. Maidana, T. Parhizkar, A. Gomola, I. B. Utne, and A. Mosleh, "Supervised dynamic probabilistic risk assessment : Review and comparison of methods," Reliab. Eng. Syst. Saf., vol. 230, no. May 2022, p. 108889, 2023, doi: 10.1016/j.ress.2022.108889.

[45] F. M. Safie, R. G. Stutts, and Z. Huang, "Reliability and probabilistic risk assessment - How they play together," in Proceedings - Annual Reliability and Maintainability Symposium, 2015, vol. 2015-May, pp. 1–5, doi: 10.1109/RAMS.2015.7105058.

[46] Q. Liu, L. Xing, and C. Wang, "Framework of Probabilistic Risk Assessment for Security and Reliability," Proc. - 2017 IEEE 2nd Int. Conf. Data Sci. Cyberspace, DSC 2017, pp. 619–624, 2017, doi: 10.1109/DSC.2017.35.

[47] F. U. Muram, M. A. Javed, and S. Punnekkat, "System of Systems Hazard Analysis Using HAZOP and FTA for Advanced Quarry Production," 2019 4th Int. Conf. Syst. Reliab. Safety, ICSRS 2019, pp. 394–401, 2019, doi: 10.1109/ICSRS48664.2019.8987613.

[48] R. A. Viegas, F. de A. da S. Mota, A. P. C. S. Costa, and F. F. P. dos Santos, "A multi-criteria-based hazard and operability analysis for process safety," Process Saf. Environ. Prot., vol. 144, pp. 310–321, 2020, doi: 10.1016/j.psep.2020.07.034.

[49] J. Hu and Y. Cao, "Fuzzy Petri net based dynamic risk analysis of complex system considering protection layers," 2015 12th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2015, pp. 308–312, 2016, doi: 10.1109/FSKD.2015.7381959.

[50] P. Grimmeisen, A. Morozov, T. Fabarisov, A. Wortmann, and C. H. Koo, "Automated Model-Based Reliability Assessment of Software-Defined Manufacturing," IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, vol. 2022-Septe, pp. 1–4, 2022, doi: 10.1109/ETFA52439.2022.9921704.

[51] D. Liu, X. Xu, K. Ma, L. Tao, and M. Suo, "Fault Diagnosis Based on Fault Tree and Bayesian Network with Grey Optimization," Proc. 34th Chinese Control Decis. Conf. CCDC 2022, pp. 1787–1792, 2022, doi: 10.1109/CCDC55256.2022.10033578.

[52] A. M. Stanciu and H. Ciocârlie, "Analyzing Code Security: Approaches and Tools for Effective Review and Analysis," Int. Conf. Electr. Comput. Energy Technol. ICECET 2023, no. November, pp. 1–6, 2023, doi: 10.1109/ICECET58911.2023.10389326.

[53] Q. Li and C. Mao, "Considering Testing-Coverage and Fault Removal Efficiency Subject to the Random Field Environments with Imperfect Debugging in Software Reliability Assessment," Proc. - 2016 IEEE 27th Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2016, pp. 257–263, 2016, doi: 10.1109/ISSREW.2016.13.

[54] C. Jackson, "Reliability growth and demonstration: The Multi-Phase Reliability Growth Model (MPRGM)," Proc. - Annu. Reliab. Maintainab. Symp., vol. 2016-April, pp. 1–6, 2016, doi: 10.1109/RAMS.2016.7447983.

[55] Z. S. Li and D. Xu, "A bayesian approach for modeling reliability growth," Proc. - Annu. Reliab. Maintainab. Symp., vol. 2020-Janua, 2020, doi: 10.1109/RAMS48030.2020.9153616.

[56] T. Sutter, T. Kehrer, M. Rennhard, B. Tellenbach, and J. Klein, "Dynamic Security Analysis on Android: A Systematic Literature Review," IEEE Access, vol. 12, no. April, pp. 57261–57287, 2024, doi: 10.1109/ACCESS.2024.3390612.

[57] P. Vats, A. Gossain, and M. Mandot, "SARLA - A 3-Tier Architectural Framework Based on the ACO for the Probablistic Analysis of the Regression Test Case Selection and their Prioritization," ICRITO 2020 - IEEE 8th Int. Conf. Reliab. Infocom Technol. Optim. (Trends Futur. Dir., pp. 681–687, 2020, doi: 10.1109/ICRITO48877.2020.9198020.

[58] D. Ling, B. Liu, and S. Wang, "A component-based software reliability assessment method considering component effective behavior," 2017 2nd Int. Conf. Reliab. Syst. Eng. ICRSE 2017, no. Icrse, 2017, doi: 10.1109/ICRSE.2017.8030748.

[59] 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.

[60] Y. Xu and S. He, "Avionics Equipment Cable Fault Injection Design and Simulation Testing Validation," 2023 Glob. Reliab. Progn. Heal. Manag. Conf. PHM-Hangzhou 2023, pp. 1–5, 2023, doi: 10.1109/PHM-HANGZHOU58797.2023.10482733.

[61] R. Ramler and C. Klammer, "Enhancing Acceptance Test-Driven Development with Model-Based Test Generation," Proc. - Companion 19th IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS-C 2019, pp. 503–504, 2019, doi: 10.1109/QRS-C.2019.00096.

[62] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of systems engineering: Basic concepts, model-based techniques, and research directions," ACM Comput. Surv., vol. 48, no. 2, 2015, doi: 10.1145/2794381.

[63] M. Jahan, Z. S. H. Abad, and B. Far, "Detecting Emergent Behavior in Scenario-Based Specifications using a Probabilistic Model," Proc. - 10th Int. Model. Requir. Eng. Work. MoDRE 2020, pp. 31–38, 2020, doi: 10.1109/MoDRE51215.2020.00010.

[64] 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.

[65] L. Fraccascia, I. Giannoccaro, and V. Albino, "Resilience of complex systems: State of the art and directions for future research," Complexity, vol. 2018, 2018, doi: 10.1155/2018/3421529.

[66] J. Zhang, Y. He, Y. Xie, and A. Zhang, "Product Key Reliability Characteristics Identification Approach Using GA in Manufacturing Process," 2021 Glob. Reliab. Progn. Heal. Manag. PHM-Nanjing 2021, 2021, doi: 10.1109/PHM-Nanjing52125.2021.9612881.

[67] B. Liao, Y. Ali, S. Nazir, L. He, and H. U. Khan, "Security Analysis of IoT Devices by Using Mobile Computing: A Systematic Literature Review," IEEE Access, vol. 8, pp. 120331–120350, 2020, doi: 10.1109/ACCESS.2020.3006358.

[68] T. S. Sakriwala, V. Pandey, and R. K. S. Raveendran, "Reliability Assessment Framework for Additive Manufactured Products," 2020 Int. Conf. Comput. Perform. Eval. ComPE 2020, pp. 350–354, 2020, doi: 10.1109/ComPE49325.2020.9200078.