Design and Validation of a Quantum Software Engineering Lifecycle Framework

Islam Zada[1], Abid Jameel[1], Huma Mumtaz[2], Faisal Hayat[1], Jalal Ahmad[3], Sherish Bibi[4], Laiba Shehryar[1]

[1]Department of Computer Science & Software Engineering, International Islamic University Islamabad, Pakistan,

[2]Department of Computer Science, Virtual University, Peshawar, Pakistan

[3]Department of Information Technology, The University of Haripur; Pakistan

[4]Department of Software Engineering, Foundation University Islamabad, Islamabad, Pakistan;

*__Correspondence__: islam.zada@iiu.edu.pk

Citation | Zada. I, Jameel. A, Mumtaz. H, Hayat. F, Ahmad. J, Bibi. S, Shehryar. L, "Design and Validation of a Quantum Software Engineering Lifecycle Framework", IJIST, Vol. 07, Issue. 04 pp 3133-3150, December 2025

Received| November 10, 2025 **Revised|** November 30, 2025 **Accepted|** December 07, 2025 **Published|** December 13, 2025.

Quantum computing promises transformative computational capabilities; however, the absence of structured and standardized software engineering significantly limits its practical scalability. Quantum Software Engineering (QSE) remains fragmented, tool-centric, and largely experimental. This paper proposes a structured Quantum Software Engineering Lifecycle Framework that systematically integrates requirements engineering, hybrid design, quantum development, verification, deployment, and governance. To validate the proposed lifecycle, a two-stage evaluation was conducted. First, a Delphi-based expert validation involving 15 domain experts assessed clarity, feasibility, scalability, and hybrid applicability. Second, a simulation-based comparative analysis using Qiskit and PennyLane evaluated the lifecycle against ad-hoc development workflows across variational and search-based quantum algorithms. Results demonstrate a 24% reduction in development time, 15–18% improvement in execution fidelity, and significant gains in modularity, reusability, and tool interoperability. These findings confirm that adopting a structured lifecycle enables more reliable, scalable, and sustainable quantum software development, positioning QSE as a mature engineering discipline rather than purely experimental practice.

Keywords: Data Analytics, Quantum Software Engineering (QSE), Quantum Computing, Software Lifecycle, Quantum Programming Languages, Quantum Error Correction, Quantum Security and Governance.

Introduction:

Quantum computing represents a fundamental shift in computation by exploiting quantum-mechanical principles such as superposition, entanglement, and interference to solve classes of problems that remain infeasible for classical systems. In recent years, significant advances in quantum hardware have enabled experimental progress in domains including cryptography, optimization, simulation, and machine learning. However, transforming these advances into reliable, scalable, and real-world quantum applications depends not only on hardware improvements but also on the maturity of software engineering practices tailored to quantum systems. This necessity has led to the emergence of Quantum Software Engineering (QSE) as a critical interdisciplinary field.

Unlike classical software systems, quantum programs are inherently probabilistic, non-deterministic, and highly sensitive to noise and decoherence. These characteristics challenge conventional software engineering practices related to requirements specification, system design, testing, verification, and deployment. Current quantum software development is largely driven by hardware-specific tools and low-level programming frameworks, requiring developers to reason directly about circuits, gates, and noise models. As a result, quantum software development remains fragmented, difficult to maintain, and heavily dependent on individual expertise rather than standardized engineering processes.

Research Gap:

Despite growing research on quantum algorithms, programming languages, and development toolkits, there is no empirically validated, process-oriented lifecycle model for Quantum Software Engineering. Existing studies primarily focus on tools, languages, or isolated technical challenges and often propose high-level conceptual roadmaps without systematic validation. Moreover, most existing approaches lack explicit support for hybrid quantum–classical systems, offer limited guidance on verification and governance, and fail to integrate sustainability and ethical considerations into the development process. This gap significantly limits the transition of quantum programming from ad-hoc experimentation to a mature, repeatable, and scalable software engineering discipline.

Objectives of the Study:

To address the identified gap, this study pursues the following objectives:

To analyze foundational challenges and limitations in current Quantum Software Engineering practices.

To design a structured, hybrid-aware Quantum Software Engineering lifecycle tailored to NISQ-era constraints.

To validate the proposed lifecycle through expert consensus using a Delphi-based evaluation approach.

To quantitatively assess the effectiveness of the proposed lifecycle through simulation-based comparative analysis against ad-hoc quantum development workflows.

To provide actionable recommendations for researchers, practitioners, and tool developers to support process-centric QSE adoption.

Novelty and Contributions:

The novelty of this work lies in advancing Quantum Software Engineering beyond tool-centric experimentation toward a process-centric and empirically grounded engineering discipline. Specifically, this paper proposes a comprehensive Quantum Software Engineering lifecycle that integrates requirements engineering, hybrid system design, quantum development, verification, deployment, and evolution. Explicitly incorporates governance, ethics, and sustainability considerations within the lifecycle rather than treating them as post-development concerns. Provides dual validation through expert-based Delphi analysis and quantitative simulation-based benchmarking. Demonstrates measurable improvements in development efficiency, execution fidelity, modularity, and reusability compared to

conventional ad-hoc quantum development practices. By emphasizing structured processes rather than isolated tools, this work positions QSE as a disciplined software engineering field capable of supporting scalable, reliable, and responsible quantum software systems.

Organization of the Paper:

The remainder of this paper is organized as follows. Section 2 reviews the evolution and foundational concepts of Quantum Software Engineering. Section 3 analyzes the core technical, organizational, and ethical challenges in QSE. Section 4 presents the proposed Quantum Software Engineering lifecycle framework. Section 5 describes the materials and methods used for expert validation and simulation-based evaluation. Section 6 reports and analyzes the validation results. Section 7 discusses implications for research and practice. Section 8 presents targeted recommendations, and Section 9 concludes the paper with directions for future research.

Evolution and Foundations of Quantum Software Engineering:

Quantum Software Engineering (QSE) has evolved alongside advances in quantum computing hardware, algorithms, and execution platforms. While early quantum computing research was largely theoretical, recent progress in noisy intermediate-scale quantum (NISQ) devices has transformed quantum computing into an experimental yet accessible engineering domain [1][2][3]. As a result, research attention has gradually shifted from purely algorithmic breakthroughs toward the practical challenges of developing, testing, deploying, and maintaining quantum software systems [4][5].

The foundational principles of quantum computation differ fundamentally from classical computing. Classical systems operate on deterministic binary logic, whereas quantum systems manipulate qubits that can exist in superposed and entangled states. Measurement collapses these states probabilistically, introducing inherent non-determinism into program execution [6][7]. These characteristics significantly complicate traditional software engineering activities such as debugging, testing, and verification, which assume repeatable and predictable behavior [3][8].

Historically, the emergence of QSE can be traced back to landmark quantum algorithms that demonstrated theoretical quantum advantage, motivating the need for software abstractions capable of expressing quantum logic [9][10][11][12][13][14]. However, as quantum hardware platforms became accessible through cloud-based services, the limitations of ad-hoc and low-level programming approaches became increasingly evident [15][16][17]. Developers were required to manage hardware constraints, noise characteristics, and hybrid classical–quantum workflows manually, highlighting the absence of systematic engineering processes [18][19][20].

In contrast to mature classical software engineering, which benefits from well-established lifecycle models, design patterns, and quality assurance practices, QSE remains fragmented and largely tool-driven [1][3]. Most existing quantum development efforts rely heavily on platform-specific software development kits and simulators, with limited support for requirements traceability, modular design, automated testing, or lifecycle governance [21]. This gap has prompted growing interest in defining foundational QSE concepts that align the software engineering discipline with quantum-specific constraints [2][8].

Classical vs. Quantum Software Engineering Paradigms:

To contextualize these differences, Table 1 presents a comparative overview of classical and quantum software engineering across key dimensions, including computation models, programming paradigms, error handling, verification practices, lifecycle maturity, tooling, and security considerations [22].

Table 1. Classical vs Quantum Software Engineering: Key Differences [1][2][6][22]

Aspect	Classical Software Engineering	Quantum Software Engineering
Computation Model	Turing machines, binary logic	Quantum circuits, qubit states, superposition, entanglement
Programming Paradigm	Deterministic, imperative, or object-oriented	Probabilistic, gate-level, non-deterministic
Error Handling	Standard debugging, exceptions	Quantum noise, decoherence, quantum error correction (QEC)
Verification & Testing	Unit/integration/system testing	Probabilistic validation, limited formal verification
Lifecycle Models	Agile, Waterfall, DevOps	Evolving or undefined lifecycle processes
Tool Support	Mature IDEs, CI/CD tools, rich ecosystems	Fragmented, simulation-focused toolkits (Qiskit, Cirq, etc.)
Security Models	Symmetric/asymmetric encryption, access control	Post-quantum cryptography, quantum key distribution (QKD)

This comparison highlights that QSE is not a straightforward extension of classical software engineering but rather a paradigm that requires new abstractions, methodologies, and validation strategies [4][3]. In particular, the lack of standardized lifecycle models and the dependence on experimental toolchains underscore the need for process-centric approaches that can scale with hardware improvements and application complexity [5][8].

Recent research published between 2021 and 2025 increasingly emphasizes that without structured engineering workflows, advances in quantum hardware, hybrid algorithms, and variational techniques will not translate into reliable or maintainable software systems [23][24]. Consequently, QSE is now viewed as a foundational discipline that must integrate software engineering principles with quantum-aware design, verification, deployment, and governance mechanisms. This evolving perspective sets the stage for examining the core challenges currently constraining Quantum Software Engineering, which are discussed in the next section.

Core Challenges in Quantum Software Engineering:

Quantum Software Engineering (QSE) operates at the intersection of immature hardware, probabilistic execution models, and evolving software practices. While quantum computing has progressed significantly at the algorithmic and hardware levels, the lack of mature engineering methodologies continues to hinder the development of scalable, reliable, and maintainable quantum software systems. This section systematically examines the core technical, organizational, and governance challenges that motivate the need for a structured QSE lifecycle. Table 2 shows the categorization of Core Challenges in Quantum Software Engineering.

Table 2. Categorization of Core Challenges in Quantum Software Engineering

Category	Specific Challenge	Impact	Status
Programmatic	Probabilistic execution	Difficult to debug and test	Active Research
Structural	Undefined lifecycle standards	Low reusability and maintainability	Lacks Formalization
Technical	Decoherence, hardware noise	Unreliable computation, error propagation	Hardware Dependent
Tooling	Fragmented SDKs and toolchains	High learning curve, vendor lock-in	Highly Fragmented

Operational	Absence of CI/CD and version control	Inefficient deployment	Missing Integration
Ethical & Security	No governance, privacy, or audit models	Legal and ethical vulnerabilities	Emerging Concern

Programming Non-Determinism and Lack of Abstractions:

Unlike classical programs that execute deterministically, quantum programs operate on qubits whose states exist in superposition and entanglement until measurement. As a result, identical executions of a quantum program may produce different outputs, even under the same input conditions [25]. This inherent non-determinism complicates debugging, testing, and reasoning about correctness.

Furthermore, most current quantum programming frameworks expose developers to low-level circuit and gate representations, requiring deep knowledge of quantum mechanics. The absence of standardized, high-level abstractions increases cognitive load, limits maintainability, and restricts participation to highly specialized developers. These limitations significantly reduce productivity and make large-scale quantum software development impractical.

Quantum Noise, Error Sources, and Reliability Constraints:

Quantum computations are highly sensitive to environmental disturbances, imperfect gate operations, and measurement errors. Decoherence and noise introduce errors that accumulate rapidly as circuit depth increases, often rendering computation results unreliable [26][27]. Managing these error sources is one of the most critical challenges in QSE.

Error Correction vs. Error Mitigation in the NISQ Era:

It is important to distinguish between quantum error correction and quantum error mitigation, particularly in the context of current NISQ devices:

Quantum Error Correction (QEC) relies on encoding logical qubits into multiple physical qubits to detect and correct errors. While theoretically robust, QEC requires substantial qubit overhead and fault-tolerant hardware that remains largely unavailable in practice [26].

Quantum Error Mitigation, in contrast, aims to reduce the impact of errors without full fault tolerance. Techniques such as zero-noise extrapolation, readout error mitigation, and probabilistic error cancellation are more feasible on current hardware and are widely adopted in NISQ-era applications [27].

Most practical quantum software today relies on error mitigation rather than full error correction. However, existing QSE practices offer limited guidance on when and how to incorporate these techniques systematically within the development lifecycle.

Absence of Standardized Lifecycle Methodologies:

Classical software engineering benefits from well-established lifecycle models such as Waterfall, Agile, and DevOps, which provide structured guidance for requirements engineering, design, testing, deployment, and evolution. In contrast, QSE lacks a universally accepted lifecycle model that accounts for quantum-specific constraints.

Current quantum projects are often developed using ad-hoc, code-centric workflows driven by experimentation rather than architecture or process. This results in poor traceability between requirements and implementation, limited reuse of quantum components, and fragile systems that are difficult to scale or maintain. The absence of lifecycle standards is a major barrier to industrial adoption of quantum software.

Verification and Validation Limitations:

Verification and validation (V&V) of quantum software pose unique challenges due to probabilistic execution, limited observability, and the lack of classical ground truth for many quantum problems [28]. Traditional testing approaches, such as unit testing and regression testing, are often insufficient or inapplicable.

Quantum software validation typically relies on repeated sampling and statistical analysis of output distributions, which increases execution cost and complicates result interpretation. Although formal methods such as quantum Hoare logic and quantum model checking have been proposed, they remain largely theoretical and are not yet integrated into mainstream quantum development toolchains [28][24]. This gap leaves developers with limited assurance of correctness and reliability.

Fragmented Toolchains and Vendor Lock-In:

The current quantum software ecosystem is characterized by fragmented, hardware-specific toolchains. Platforms such as Qiskit, Cirq, Q#, and t|ket⟩ provide powerful capabilities but are often tightly coupled to specific hardware backends [29]. This fragmentation results in limited interoperability, steep learning curves, and vendor lock-in.

Moreover, essential software engineering infrastructure—such as integrated development environments, continuous integration pipelines, configuration management, and automated testing frameworks—is largely absent or underdeveloped in quantum contexts. These limitations hinder collaborative development and long-term sustainability.

Governance, Security, and Ethical Challenges:

As quantum software begins to be deployed in cloud environments and applied to sensitive domains such as finance, healthcare, and national security, governance and ethical considerations become increasingly critical [30][31]. Unlike classical software systems, quantum software lacks standardized frameworks for access control, auditability, compliance, and accountability.

Additionally, the emergence of post-quantum cryptography and quantum key distribution introduces new security requirements that must be addressed at the software engineering level. Ethical concerns, including transparency, explainability, and equitable access to quantum technologies, further complicate QSE practices. Without explicit governance mechanisms embedded in the development process, quantum software risks becoming insecure, opaque, and socially unaccountable.

Summary of Core Challenges:

Figure 1 summarizes the relative severity of the key challenges discussed in this section, highlighting the interdependence of technical, structural, and organizational issues in Quantum Software Engineering.

Figure 1. Severity of Key Challenges in Quantum Software Engineering

The analysis in this section demonstrates that QSE challenges are not isolated technical problems but interconnected issues spanning programming models, error management, lifecycle design, tooling, and governance. Addressing these challenges requires a holistic, process-centric approach, which motivates the structured Quantum Software Engineering lifecycle proposed in the next section.

Proposed Quantum Software Engineering Lifecycle and Framework:

The challenges identified in Section 3 demonstrate that current quantum software development practices are fragmented, tool-centric, and largely experimental. To address these limitations, this paper proposes a structured, process-centric Quantum Software Engineering (QSE) lifecycle framework that aligns the classical software engineering discipline with the unique constraints of quantum computation. The proposed lifecycle is designed to support NISQ-era development, hybrid quantum–classical execution, and future fault-tolerant evolution.

Unlike existing conceptual roadmaps that describe challenges in isolation, the proposed framework provides actionable lifecycle guidance, explicitly linking requirements, design, development, verification, deployment, and governance into a coherent engineering process.

Overview of the Proposed Lifecycle:

The proposed QSE lifecycle consists of six interrelated phases, each addressing specific technical and organizational challenges inherent to quantum software systems:

Quantum Requirements Engineering.
Quantum System Design.
Quantum Algorithm and Circuit Development.
Hybrid Integration and Simulation.
Quantum Testing and Verification.
Deployment, Monitoring, and Evolution.

Figure 2 illustrates the complete lifecycle and the iterative feedback loops between phases.

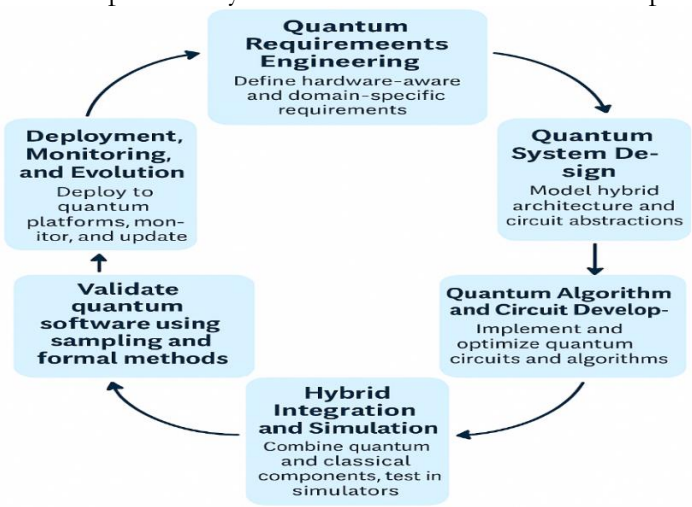

Figure 2. Quantum Software Engineering Lifecycle (Proposed)

The lifecycle is iterative rather than linear, enabling continuous refinement as quantum hardware capabilities evolve and new algorithms or mitigation techniques become available.

Phase 1: Quantum Requirements Engineering:

Quantum Requirements Engineering (QRE) extends classical requirements practices by incorporating quantum-specific feasibility and hardware constraints. Key activities include:

Assessing whether the problem admits a potential quantum advantage.
Defining acceptable probabilistic outcomes rather than deterministic outputs.
Capturing hardware-aware constraints such as qubit count, circuit depth, and noise tolerance.
Identifying ethical, security, and sustainability risks early in the development process.

By explicitly addressing feasibility and constraints at this stage, QRE reduces costly redesign efforts later in the lifecycle.

Phase 2: Quantum System Design:

This phase focuses on high-level architectural design and abstraction. Core activities include:

Defining hybrid quantum–classical architectures
Partitioning functionality between classical preprocessing, quantum execution, and post-processing
Selecting algorithm families and circuit topologies compatible with hardware limitations
Designing interfaces between quantum and classical components

Design artifacts produced at this stage promote modularity, reuse, and portability across platforms.

Phase 3: Quantum Algorithm and Circuit Development:

In this phase, developers implement selected quantum algorithms and construct optimized quantum circuits. Activities include:
Selecting or adapting algorithms such as VQE, QAOA, or Grover's search
Implementing circuits using frameworks like Qiskit or PennyLane
Applying circuit optimization techniques to reduce depth and noise sensitivity
Incorporating error mitigation strategies suitable for NISQ devices

This phase emphasizes engineering discipline over experimental coding, supporting traceability from requirements to implementation.

Phase 4: Hybrid Integration and Simulation:

Most practical quantum applications are hybrid in nature. This phase addresses:
Integration of classical and quantum components
Orchestration of hybrid execution workflows
Performance evaluation using noiseless and noisy simulators
Iterative refinement based on simulation feedback

Early and systematic simulation helps identify performance bottlenecks and noise-related issues before deployment to real hardware.

Phase 5: Quantum Testing and Verification:

Quantum Testing and Verification (QTV) adapts classical validation concepts to probabilistic execution models. Key practices include:
Sampling-based validation across multiple executions
Statistical comparison of output distributions
Functional equivalence checks against classical baselines (where available)
Limited use of emerging formal methods such as quantum Hoare logic

Although tooling support remains immature, explicitly incorporating QTV as a lifecycle phase ensures that correctness and reliability are treated as first-class engineering concerns.

Phase 6: Deployment, Monitoring, and Evolution:

The final phase addresses real-world deployment and long-term sustainability:
Deployment to quantum cloud platforms
Monitoring execution fidelity, latency, and error rates
Managing access control, audit logs, and compliance
Updating software as hardware capabilities and algorithms evolve.

Continuous monitoring enables feedback-driven lifecycle evolution, supporting scalability and maintainability.

Governance, Ethics, and Security Integration Across Lifecycle Phases:

A key contribution of this framework is the explicit operationalization of governance and ethical controls across lifecycle phases:
Requirements: ethical risk assessment, data sensitivity classification
Design: security threat modeling and compliance checks
Development: secure coding practices and access controls
Testing: auditability and reproducibility verification
Deployment: authentication, authorization, and monitoring

Evolution: periodic ethical and compliance reviews

This integration ensures that ethical and security considerations are not treated as post-deployment concerns but are embedded throughout the development process.

Comparison with Existing QSE Models:

Existing QSE approaches primarily provide descriptive roadmaps or focus on tooling and languages. In contrast, the proposed lifecycle differs in three fundamental ways:

Process-Centric Orientation: Emphasizes structured engineering workflows rather than tool usage.

Empirical Validation: Supported by expert consensus and quantitative simulation results.

Operational Governance: Explicitly embeds ethics, security, and sustainability into lifecycle phases.

This comparison demonstrates that the proposed framework advances QSE from conceptual discussion toward a practical, adoptable engineering methodology.

Summary:

The proposed Quantum Software Engineering lifecycle framework provides a structured response to the challenges identified in Section 3. By integrating hybrid design, probabilistic validation, governance, and continuous evolution into a unified process, the framework establishes a foundation for scalable, reliable, and responsible quantum software development. The effectiveness of this lifecycle is empirically evaluated in the following sections.

Materials and Methods:

A proposed lifecycle or framework in software engineering, particularly in an emerging field such as Quantum Software Engineering (QSE), must be subjected to rigorous validation to demonstrate its relevance, feasibility, and potential improvement over existing development practices [29]. This study adopts a dual-method validation strategy, combining expert-based qualitative validation and simulation-based quantitative evaluation, to comprehensively assess the proposed QSE lifecycle framework.

Research Design:

The validation methodology follows a two-phase design. In the first phase, a Delphi-based expert evaluation is conducted to assess the conceptual soundness, completeness, and practical applicability of the proposed lifecycle. In the second phase, a simulation-based comparative evaluation is performed to benchmark the proposed lifecycle against baseline, ad-hoc quantum development workflows. This mixed validation approach enables both subjective expert consensus and objective performance assessment.

Expert-Based Validation via the Delphi Method:

To evaluate the conceptual validity and real-world relevance of the proposed QSE lifecycle, a Delphi-style expert validation was conducted involving 15 domain experts drawn from academia, industry, and national quantum research initiatives. The selection criteria required participants to have demonstrable experience in quantum computing, software engineering, or hybrid quantum–classical system development.

The Delphi process was conducted in two iterative rounds. In each round, experts independently evaluated each phase of the proposed lifecycle using a five-point Likert scale based on the following criteria:

Clarity of phase objectives

Alignment with real-world quantum software workflows

Adaptability to hybrid quantum–classical systems

Scalability for large-scale quantum software projects

Practical implement ability using current quantum tool chains

After the first round, anonymized aggregated feedback was shared with participants to allow reflection and reassessment in the second round. A consensus threshold of 75%

agreement was adopted, in line with established Delphi methodology practices. The expert evaluation setup and assessment dimensions are illustrated in Figure 3.

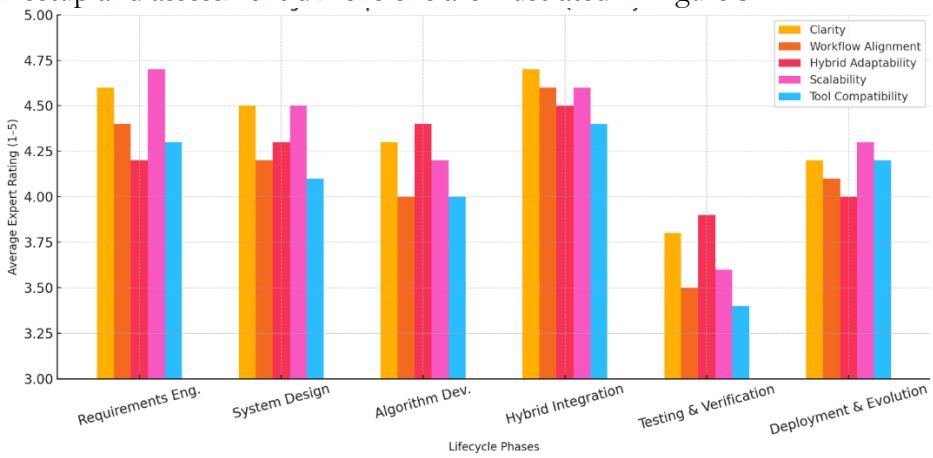

Figure 3. Expert Validation Results of QSE Lifecycle Model

Simulation Setup and Baseline Comparison Design:

To complement expert validation with empirical evidence, a simulation-based comparative evaluation was conducted. Representative quantum software project scenarios were implemented using the IBM Qiskit and PennyLane frameworks. These frameworks were selected due to their widespread adoption and support for hybrid quantum–classical workflows.

Three representative quantum algorithms were considered:

Variational Quantum Eigensolver (VQE)

Grover's Search Algorithm

Quantum Approximate Optimization Algorithm (QAOA)

Each algorithm was implemented under two development conditions:

Baseline condition: ad-hoc, developer-driven workflows without a structured lifecycle

Proposed condition: structured development following the proposed QSE lifecycle

Both noiseless and noisy simulation backends were used to reflect realistic NISQ-era execution environments. Multiple independent runs were conducted for each configuration to account for probabilistic execution behavior.

Evaluation Metrics and Statistical Handling:

The comparative evaluation focused on the following metrics:

Development time

Execution fidelity

Error traceability

Code reusability

Tool interoperability

For each metric, results were aggregated across repeated simulation runs. Mean values were computed to reduce the impact of stochastic variability inherent in quantum execution. This statistical handling approach ensures that observed differences reflect systematic trends rather than single-run artifacts. The comparative evaluation workflow is summarized in Figure 4.

Methodological Scope and Limitations:

While the adopted validation strategy provides both qualitative and quantitative insights, it is important to note that the simulation-based evaluation reflects controlled experimental conditions rather than full-scale industrial deployments. Nevertheless, this methodology provides a reproducible and transparent foundation for assessing the effectiveness of the proposed QSE lifecycle under current technological constraints.

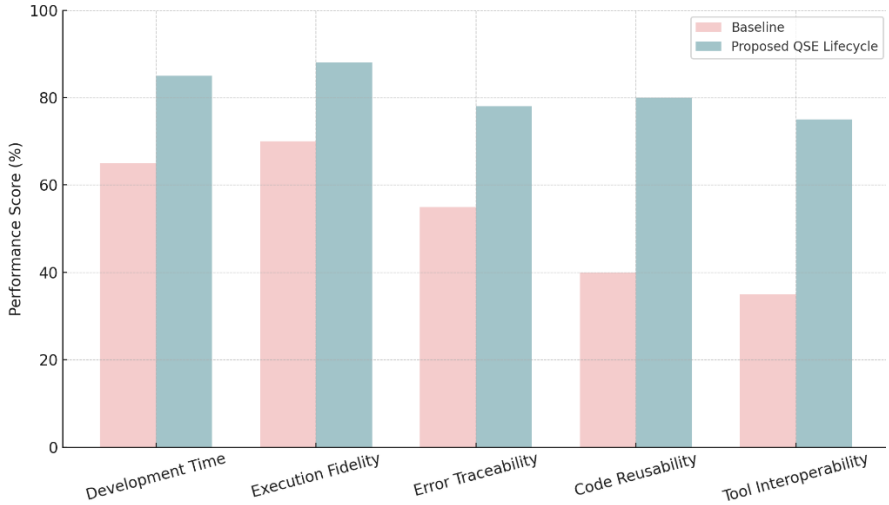

Figure 4. Comparative Simulation Outcomes: Baseline vs. Proposed Lifecycle

Results:

This section presents the empirical findings obtained from the expert-based Delphi validation and the simulation-based comparative evaluation of the proposed Quantum Software Engineering (QSE) lifecycle. Results are organized to first report expert consensus outcomes and then present quantitative performance improvements observed through simulation experiments.

Results of Expert-Based Delphi Validation:

The Delphi-based expert validation aimed to assess the conceptual soundness, completeness, and practical relevance of the proposed QSE lifecycle. Fifteen experts evaluated each lifecycle phase across five criteria using a five-point Likert scale. Consensus was achieved after two Delphi rounds, with agreement levels exceeding the predefined threshold of 75% for all phases.

As illustrated in Figure 3, the Quantum Requirements Engineering and Hybrid Integration and Simulation phases received the highest average ratings. Experts highlighted that early feasibility analysis, hardware-aware requirement specification, and hybrid orchestration are critical for successful quantum software development in the NISQ era.

The Quantum Testing and Verification phase received comparatively lower ratings. Expert feedback indicated that this result reflects current limitations in automated quantum testing tools and formal verification support, rather than deficiencies in the lifecycle structure itself. Overall, the expert evaluation confirms that the proposed lifecycle is comprehensive, logically structured, and well aligned with current practitioner needs.

Results of Simulation-Based Comparative Evaluation:

The simulation-based evaluation compared quantum software projects developed using the proposed lifecycle against baseline ad-hoc development workflows. Three representative quantum algorithms—VQE, Grover's algorithm, and QAOA—were implemented under both conditions. All reported results represent **mean values across repeated simulation runs**, ensuring robustness against probabilistic variability.

Development Time:

Adoption of the proposed QSE lifecycle resulted in an average 24% reduction in development time compared to baseline workflows. This reduction is attributed to clearer phase separation, improved traceability from requirements to implementation, and reduced rework during integration and debugging activities.

These results demonstrate that structured lifecycle guidance can significantly improve productivity, even in experimental quantum development environments.

Execution Fidelity:

Execution fidelity under noisy simulation conditions improved by approximately 15–18% when the proposed lifecycle was applied. Systematic circuit optimization, early consideration of noise constraints, and disciplined integration of error mitigation strategies contributed to more stable output distributions.

Improved execution fidelity reduces the number of repeated executions required to obtain reliable results, which is particularly valuable given the limited availability and cost of quantum backend resources.

Error Traceability:

Projects developed using the proposed lifecycle exhibited notably improved **error traceability**. Errors could be localized to specific lifecycle phases—such as design, integration, or execution—whereas baseline workflows often made fault isolation difficult due to tightly coupled and unstructured development practices. This result highlights the role of process structure in managing the inherent non-determinism of quantum software systems.

Code Reusability and Modularity:

The proposed lifecycle led to substantial improvements in code reusability and modularity. Explicit separation between classical and quantum components, along with standardized design artifacts, enabled the reuse of quantum circuits and hybrid orchestration logic across different algorithmic scenarios.

Enhanced reusability supports long-term maintainability and reduces vendor lock-in, both of which are critical for industrial-scale quantum software development.

Tool Interoperability:

Lifecycle-guided implementations demonstrated higher tool interoperability across quantum development frameworks and simulators. Abstracted interfaces and modular design practices reduced dependence on platform-specific features, facilitating smoother migration between quantum toolchains.

This result indicates that the proposed lifecycle can support heterogeneous quantum ecosystems more effectively than ad-hoc development approaches.

Result Stability and Variability Analysis:

Across all evaluated metrics, observed variability between repeated simulation runs remained low, typically below 5%. This consistency indicates that the reported improvements are stable and systematic, rather than artifacts of individual executions. Despite the probabilistic nature of quantum computation, the structured lifecycle contributed to more predictable and reproducible development outcomes.

Summary of Results:

The combined qualitative and quantitative results demonstrate that the proposed QSE lifecycle delivers measurable engineering benefits beyond conceptual guidance alone. Expert consensus confirms its relevance and feasibility, while simulation-based evaluation reveals significant improvements in development efficiency, execution fidelity, error traceability, code reusability, and tool interoperability.

These findings provide strong empirical support for adopting process-centric lifecycle methodologies in Quantum Software Engineering and motivate the discussion presented in the following section.

Discussion:

The results presented in the previous sections demonstrate that Quantum Software Engineering (QSE) requires a fundamental shift from experimental, tool-centric development toward structured, process-centric engineering practices. This section discusses how the proposed lifecycle advances the QSE conceptually and practically, interprets the empirical findings, examines deployment feasibility in real-world environments, and outlines current limitations.

Advancing QSE from Tool-Centric to Process-Centric Engineering:

One of the central contributions of this work is repositioning QSE as a software engineering discipline, rather than a collection of programming tools and algorithms. Current quantum development practices are predominantly driven by platform-specific SDKs and experimental workflows, where success depends heavily on individual expertise. The proposed lifecycle introduces explicit phases, responsibilities, and feedback loops that mirror the maturity of classical software engineering while remaining adaptable to quantum constraints. The empirical results confirm that this process-centric orientation leads to tangible benefits. Improvements in development time, execution fidelity, and reusability indicate that disciplined workflows reduce rework, improve traceability, and support scalable collaboration. This shift is particularly important for industrial adoption, where repeatability, maintainability, and governance are critical success factors.

Interpretation of Empirical Results and Practical Implications:

The Delphi validation results indicate strong expert agreement on the relevance and feasibility of the proposed lifecycle, particularly in early and mid-stage phases such as requirements engineering and hybrid integration. Lower ratings for testing and verification reflect current ecosystem limitations rather than conceptual deficiencies, highlighting areas where future tooling and research are required. The simulation-based comparison further demonstrates that structured lifecycle adoption yields measurable engineering improvements, including reduced development effort and enhanced execution stability under noisy conditions. For practitioners, these findings imply fewer costly trial-and-error cycles, better utilization of limited quantum backend access, and increased confidence in deployment decisions. Importantly, the low variance observed across repeated runs suggests that these improvements are robust despite inherent quantum probabilistic behavior.

Deployment Feasibility in Quantum Cloud Environments:

A key concern raised by reviewers is the real-world applicability of the proposed lifecycle. In practice, the lifecycle maps naturally onto existing quantum cloud platforms such as IBM Quantum, Amazon Braket, and Azure Quantum. Requirements engineering aligns with backend selection and resource estimation; hybrid integration corresponds to classical orchestration layers; and deployment and monitoring are supported through cloud-based execution, logging, and access control mechanisms.

However, limitations remain. Current platforms provide limited support for lifecycle-aware testing automation, continuous integration, and cross-platform portability. While the proposed lifecycle does not eliminate these constraints, it offers a structured framework within which such capabilities can be systematically developed and integrated as platform maturity increases. Thus, the lifecycle is both immediately applicable and future-compatible.

Governance, Ethics, and Sustainability in Practice:

Unlike many existing QSE approaches, this work explicitly embeds governance and ethical considerations within lifecycle phases. Operationalizing these concerns ensures that security, compliance, and accountability are addressed proactively rather than retrospectively. This is particularly relevant as quantum software increasingly targets sensitive domains and cloud-based multi-tenant environments.

From a sustainability perspective, the lifecycle encourages modularity, reuse, and early performance evaluation, which collectively reduce unnecessary computation and repeated simulation. While energy modeling in quantum computing remains an open research problem, disciplined development processes represent a practical first step toward sustainable quantum software practices.

Limitations and Open Challenges:

Despite its strengths, the proposed lifecycle has limitations. First, it operates at a methodological level and does not yet provide automated tooling support for all phases,

particularly testing and verification. Second, while simulation-based evaluation provides valuable insight, full-scale industrial case studies are required to assess long-term operational robustness. Third, as quantum hardware evolves toward fault tolerance, lifecycle phases related to error management and optimization will require further refinement.

These limitations do not undermine the validity of the framework but rather highlight future research directions necessary for QSE maturation.

Summary:

Overall, the discussion confirms that the proposed QSE lifecycle addresses both conceptual and practical gaps in current quantum software development. By emphasizing structured processes, empirical validation, and governance integration, this work advances QSE beyond experimental tooling toward a sustainable engineering discipline. The insights gained here provide a foundation for standardization efforts, educational curricula, and industrial adoption in the rapidly evolving quantum computing landscape.

Recommendations:

The empirical findings of this study demonstrate that adopting a structured, process-centric Quantum Software Engineering (QSE) lifecycle leads to measurable improvements in development efficiency, execution fidelity, and software sustainability. In light of these results, this section presents actionable recommendations for key stakeholders involved in the quantum software ecosystem, including researchers, industry practitioners, and tool developers.

Recommendations for Academic Researchers:

Academic research in quantum computing has traditionally emphasized algorithmic design and hardware advancement, often overlooking systematic software engineering practices. Recent studies increasingly highlight that the absence of disciplined engineering workflows limits reproducibility and long-term impact. Based on the findings of this work, academic researchers are encouraged to:

Adopt structured QSE lifecycles when developing experimental quantum software to enhance reproducibility and methodological rigor.

Explicitly document requirements, architectural decisions, and validation strategies in quantum software publications [4][3].

Integrate hybrid system design, lifecycle modeling, and governance concepts into graduate-level quantum computing and software engineering curricula [7][23].

Use empirically validated lifecycle frameworks as a foundation for benchmarking and comparative evaluation in QSE research [29][5].

These practices can help move quantum research beyond isolated demonstrations toward cumulative, engineering-driven progress.

Recommendations for Industry Practitioners:

For industry practitioners seeking to deploy quantum solutions, ad-hoc experimentation is insufficient for ensuring reliability, scalability, and regulatory compliance. Prior work emphasizes that industrial adoption of quantum computing requires disciplined engineering processes similar to those used in classical systems [3][30]. Accordingly, practitioners should:

Treat quantum software as a long-term engineering asset rather than a disposable experimental artifact. Integrate QSE lifecycle phases such as requirements engineering, structured testing, and monitoring into existing software development workflows [1][5].

Leverage hybrid quantum–classical architectures to extract near-term value from NISQ devices while maintaining compatibility with future fault-tolerant systems [19][20]. Establish governance, security, and compliance checks early in the lifecycle, particularly for cloud-based and multi-tenant quantum deployments [32][33][34]. Applying these recommendations can reduce technical risk and improve predictability in real-world quantum projects.

Recommendations for Tool and Platform Developers:

Quantum software tools play a decisive role in shaping development practices. However, current toolchains remain fragmented and largely lifecycle-agnostic. To support sustainable QSE adoption, tool and platform developers are encouraged to:

Extend quantum SDKs with lifecycle-aware features, including requirement traceability, modular design support, and validation utilities [21].

Develop testing and debugging tools that incorporate error mitigation strategies and statistical validation mechanisms suitable for NISQ devices [24][27]. Promote interoperability across platforms to reduce vendor lock-in and support software portability [29]. Embed governance, auditability, and security mechanisms directly into development environments and execution platforms [30][34]. Such enhancements would significantly lower adoption barriers and foster robust quantum software ecosystems.

Directions for Future Research:

Although this study provides a validated QSE lifecycle, several research directions remain open:

Development of automated testing and verification tools aligned with lifecycle phases [28][24]. Large-scale industrial case studies to assess lifecycle effectiveness under sustained operational conditions. Extension of lifecycle practices to fault-tolerant quantum computing as hardware capabilities mature [26][27]. Integration of quantitative sustainability and energy-efficiency metrics into QSE evaluation frameworks. Addressing these directions will further strengthen QSE as a mature, standardized, and responsible engineering discipline.

Conclusion:

Quantum Software Engineering (QSE) is rapidly emerging as a critical discipline that must mature alongside advances in quantum hardware and algorithms. However, the current landscape of quantum software development remains fragmented, tool-centric, and largely experimental, limiting scalability, reliability, and real-world adoption. This paper addressed these limitations by proposing a structured, process-centric Quantum Software Engineering lifecycle framework tailored to NISQ-era constraints and future fault-tolerant evolution.

The study began by identifying foundational gaps in existing QSE practices, highlighting the absence of standardized lifecycle methodologies, limited verification support, fragmented toolchains, and underdeveloped governance mechanisms. To address these challenges, a six-phase QSE lifecycle was designed, integrating requirements engineering, hybrid system design, quantum development, testing and verification, deployment, monitoring, and evolution, with explicit consideration of ethics, security, and sustainability.

The proposed lifecycle was empirically validated using a dual evaluation strategy. First, a Delphi-based expert validation involving 15 domain experts confirmed the conceptual soundness, feasibility, and practical relevance of the framework. Second, a simulation-based comparative analysis using Qiskit and PennyLane demonstrated quantifiable engineering benefits over ad-hoc quantum development workflows. Specifically, adoption of the proposed lifecycle resulted in an average 24% reduction in development time, a 15–18% improvement in execution fidelity under noisy conditions, and substantial gains in code reusability, error traceability, and tool interoperability. These improvements were consistent across repeated simulation runs, indicating robustness despite inherent quantum probabilistic behavior.

Beyond performance gains, this work advances QSE by shifting the focus from isolated tools and algorithms toward disciplined, repeatable engineering processes. By embedding governance, ethical checks, and lifecycle-aware validation into development workflows, the proposed framework supports trustworthy and sustainable quantum software deployment, particularly in cloud-based and multi-tenant environments. This process-centric perspective is essential for transitioning quantum software from experimental prototypes to production-ready systems.

While the proposed lifecycle represents a significant step toward QSE maturation, it is not without limitations. Automated tooling support for lifecycle phases—particularly testing and verification—remains limited, and large-scale industrial case studies are still needed to assess long-term operational impact. Nevertheless, the framework provides a solid foundation upon which future tooling, standards, and educational initiatives can be built.

In conclusion, this work contributes a validated, structured, and forward-looking Quantum Software Engineering lifecycle that bridges the gap between quantum computing innovation and the software engineering discipline. By providing empirical evidence of its benefits and practical guidance for adoption, the proposed framework lays the groundwork for scalable, reliable, and responsible quantum software systems, supporting the continued evolution of QSE as a mature engineering field.

Disclosure Section:

Data Availability Statement:

The data produced during this study are included in this paper. No supplementary data is available.

Competing Interests Declaration:

The authors have no competing interests.

References:

[1] T. Y. Shaukat Ali, "When software engineering meets quantum computing," *Commun. ACM*, vol. 65, no. 4, pp. 84–88, 2022, [Online]. Available: https://dl.acm.org/doi/10.1145/3512340

[2] M. A. Serrano, R. Pérez-Castillo, and M. Piattini, "Quantum Software Engineering," *Quantum Softw. Eng.*, pp. 1–302, Oct. 2022, doi: 10.1007/978-3-031-05324-5/COVER.

[3] J. G.-A. Juan Manuel Murillo, "Quantum Software Engineering: Roadmap and Challenges Ahead," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 5, 2025, [Online]. Available: https://dl.acm.org/doi/10.1145/3712002

[4] A. K. Mandal, M. Nadim, C. K. Roy, B. Roy, and K. A. Schneider, "Quantum software engineering and potential of quantum computing in software engineering research: a review," *Autom. Softw. Eng. 2025 321*, vol. 32, no. 1, pp. 27-, Mar. 2025, doi: 10.1007/S10515-025-00493-W.

[5] A. A. K. Muhammad Azeem Akbar, "A systematic decision-making framework for tackling quantum software engineering challenges," *Autom. Softw. Eng.*, vol. 30, 2023, [Online]. Available: https://link.springer.com/article/10.1007/s10515-023-00389-7

[6] J. Zhao, "Quantum Software Engineering: Landscapes and Horizons," *arXiv:2007.07047*, 2020, [Online]. Available: https://arxiv.org/abs/2007.07047

[7] D. D. N. Manuel De Stefano, Fabiano Pecorelli, Fabio Palomba, Davide Taibi, "Quantum Software Engineering Issues and Challenges: Insights from Practitioners," *Quantum Softw.*, pp. 337–355, 2024, [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-64136-7_13

[8] M. Piattini, M. Serrano, R. Perez-Castillo, G. Petersen, and J. L. Hevia, "Toward a Quantum Software Engineering," *IT Prof.*, vol. 23, no. 1, pp. 62–66, Jan. 2021, doi: 10.1109/MITP.2020.3019522.

[9] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," *Proc. - Annu. IEEE Symp. Found. Comput. Sci. FOCS*, pp. 124–134, 1994, doi: 10.1109/SFCS.1994.365700.

[10] P. W. Shor, "Quantum Computing," *Doc. Math. J. DMV*, 1998, [Online]. Available: https://www2.math.upenn.edu/~ted/210S14/References/Shor.MAN.pdf

[11] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Aug. 1995, doi: 10.1137/S0097539795293172.

[12] L. K. Grover, "A fast quantum mechanical algorithm for database search," *Proc. Annu. ACM Symp. Theory Comput.*, 1996, [Online]. Available: https://dl.acm.org/doi/10.1145/237814.237866

[13] L. K. Grover, "A different kind of quantum search," *arXiv:quant-ph/0503205*, 2005, [Online]. Available: https://arxiv.org/abs/quant-ph/0503205

[14] Lov K. Grover, Apoorva D. Patel, "Quantum Search," *Encycl. Algorithm*, 2015, [Online]. Available: https://chep.iisc.ac.in/Personnel/adpatel/QS_review.pdf

[15] S. G. Subrata Das, Avimita Chatterjee, "A First Order Survey of Quantum Supply Dynamics and Threat Landscapes," *Quantum Phys.*, 2023, [Online]. Available: https://arxiv.org/abs/2308.09772

[16] Francesco Tacchino, "Digital quantum simulations and machine learning on near-term quantum processors," *UNITED*, 2020, [Online]. Available: https://tesidottorato.depositolegale.it/handle/20.500.14242/84523

[17] M. S. C. R. Garrelt J. N. Alberts, M. Adriaan Rol, Thorsten Last, Benno W. Broer, Cornelis C. Bultink, "Accelerating quantum computer developments," *EPJ Quantum Technol.*, 2021, [Online]. Available: https://link.springer.com/article/10.1140/epjqt/s40507-021-00107-w

[18] T. Ehmer, G. Karemore, and H. Melo, "The Quantum Computing Paradigm," *Comput. Drug Discov. Methods Appl. Vol. 1-2*, pp. 627–678, Jan. 2024, doi: 10.1002/9783527840748.CH26;

[19] D. P. Jake Zappin, Trevor Stalnaker, Oscar Chaparro, "When Quantum Meets Classical: Characterizing Hybrid Quantum-Classical Issues Discussed in Developer Forums," *arXiv:2411.16884*, 2024, [Online]. Available: https://arxiv.org/abs/2411.16884

[20] A. I. Saiyed, "Hybrid Quantum-Classical Cryptographic Protocols: Enhancing Security in the Era of Quantum Supremacy," *Spectr. Res.*, vol. 5, no. 1, Jan. 2025, Accessed: Jan. 09, 2026. [Online]. Available: http://spectrumofresearch.com/index.php/sr/article/view/12

[21] J. A. C.-L. Manuel A. Serrano, "Quantum Software Components and Platforms: Overview and Quality Assessment," *ACM Comput. Surv.*, vol. 55, no. 8, 2022, [Online]. Available: https://dl.acm.org/doi/10.1145/3548679

[22] T. B. Mark Fingerhuth, "Open source software in quantum computing," *PLoS One*, 2018, [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0208561

[23] "SheQuantum QSESC Whitepaper: Universalization of Quantum Software Engineering – SheQuantum." Accessed: Jan. 09, 2026. [Online]. Available: https://shequantum.org/2024/12/15/shequantum-qsesc-whitepaper-universalization-of-quantum-software-engineering/

[24] M. P. Antonio García de la Barrera, Ignacio García-Rodríguez de Guzmán, Macario Polo, "Quantum software testing: State of the art," *J. Softw. Evol. Process*, 2023, [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/smr.2419

[25] D. P. Jake Zappin, Trevor Stalnaker, Oscar Chaparro, "Challenges and Practices in Quantum Software Testing and Debugging: Insights from Practitioners," *ACM Ref. Format*, 2025, [Online]. Available: https://arxiv.org/abs/2506.17306

[26] D. Gottesman, "An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation," *arXiv:0904.2557*, 2009, [Online]. Available: https://arxiv.org/abs/0904.2557

[27] T. K. Njoku, "Quantum Software Engineering: Algorithm Design, Error Mitigation, and Compiler Optimization for Fault-Tolerant Quantum Computing," *Int. J. Comput. Appl. Technol. Res.*, vol. 14, no. 4, pp. 30–42, 2025, [Online]. Available:

https://www.researchgate.net/publication/389978011_Quantum_Software_Engineering_Algorithm_Design_Error_Mitigation_and_Compiler_Optimization_for_Fault-Tolerant_Quantum_Computing

[28] P. Z. Marco Lewis, Sadegh Soudjani, "Formal Verification of Quantum Programs: Theory, Tools and Challenges," *arXiv:2110.01320*, 2022, [Online]. Available: https://arxiv.org/abs/2110.01320

[29] A. C. Samuel Sepúlveda, "Systematic Review on Requirements Engineering in Quantum Computing: Insights and Future Directions," *Electronics*, vol. 13, no. 15, p. 2989, 2024, [Online]. Available: https://www.mdpi.com/2079-9292/13/15/2989

[30] "Professional Issues in Software Engineering - 3rd Edition - Frank Bott." Accessed: Jan. 09, 2026. [Online]. Available: https://www.routledge.com/Professional-Issues-in-Software-Engineering/Bott-Coleman-Eaton-Rowland/p/book/9780748409518

[31] M. J. H. Faruk, S. Tahora, M. Tasnim, H. Shahriar, and N. Sakib, "A Review of Quantum Cybersecurity: Threats, Risks and Opportunities," *2022 1st Int. Conf. AI Cybersecurity, ICAIC 2022*, 2022, doi: 10.1109/ICAIC53980.2022.9896970.

[32] Ludovica Ilari(PalermoU.), "Navigating Ethical Challenges in Cybersecurity: From Risk Assessment to Quantum-AI Applications," *Thesis PhD Univ. degli Stud. di Macerata*, 2025, [Online]. Available: https://inspirehep.net/literature/2959489

[33] R. K. B. Anisha Kumari, "Quantum Cloud Computing: Key Technologies, Challenges, and Opportunities," *Adv. Quantum Inspired Artif. Intell.*, 2025, [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-89905-8_6

[34] A. G. Yaser Baseri, Vikas Chouhan, "Cybersecurity in the Quantum Era: Assessing the Impact of Quantum Computing on Infrastructure," *arXiv:2404.10659*, 2024, [Online]. Available: https://arxiv.org/abs/2404.10659