*applied sciences*

MDPI

*Article*

# QC-MM: A Metadata and Schema Model for Traceable Quantum-Circuit Experiments

**Nawel Huenchuleo [1], Samuel Sepúlveda [1,*] and Alejandro Fernández [2]**

[1] Departmento de Ciencias de la Computación e Informática, QuantumLab-DCI, Universidad de La Frontera, Temuco 4811230, Chile; n.huenchuleo01@ufromail.cl
[2] LIFIA, Facultad de Informática/CICPBA, Universidad Nacional de La Plata, La Plata, Buenos Aires 1900, Argentina; casco@lifia.info.unlp.edu.ar
* Correspondence: samuel.sepulveda@ufrontera.cl

**Abstract**

**Context:** Modern quantum-computing experimentation generates heterogeneous, context-dependent execution data whose scientific value depends on preserving calibration state, compilation decisions, and run outcomes in a traceable and repository-ready form. In the NISQ era, probabilistic outputs, time-varying hardware conditions, and opaque transpilation pipelines create a data-management problem that directly affects reproducibility, traceability, and long-term reuse of experimental records. **Goal:** This paper aims to address this gap by proposing a specialized metadata and schema model for managing quantum-circuit execution data as governed, machine-interpretable, and evolvable repository artifacts. **Proposal:** We propose QC-MM, a platform-agnostic metadata model for capturing, validating, and relating contextual evidence of quantum-circuit experiments. The model integrates time-indexed calibration binding, transpilation traceability, lightweight provenance links, validation rules, and controlled schema evolution through a JSON Schema specification. **Results:** The evaluation follows a multi-scenario protocol and shows that QC-MM captures dynamic calibration context in IBM Quantum Cloud, remains interoperable through a local SpinQ NMR device, and makes transpilation effects traceable through structured records. It also supports repeated-run statistical reporting and links compilation decisions to execution outcomes, including circuit-depth reductions and changes in an estimated fidelity proxy under different optimization settings. **Conclusions:** QC-MM provides a specialized data-modeling and schema-governance foundation for traceable quantum-experiment repositories. Beyond improving reproducibility-oriented reporting, the proposal contributes to metadata validation, controlled schema evolution, and repository-oriented management of contextual experimental data.

**Keywords:** metadata schema evolution; data modeling; traceability; provenance; quantum circuit execution; quantum experiment repositories; JSON schema; QC-MM



Academic Editor: Nikos Konofaos

Received: 15 May 2026
Revised: 22 June 2026
Accepted: 23 June 2026
Published: 24 June 2026



## 1. Introduction

Modern experimental computing environments increasingly depend on the ability to store, govern, validate, and reuse complex data generated across heterogeneous platforms. In this context, advanced data practices are expected not only to persist outputs but also to preserve the contextual information required to interpret them correctly over time. This requirement is particularly acute in emerging computational domains, where data are produced under evolving infrastructures, platform-specific constraints, and non-stationary execution conditions. Quantum computing (QC) is one such domain.

QC departs substantially from classical computing by relying on qubits and quantum mechanical phenomena such as superposition, entanglement, and interference [1]. These properties give rise to new computational capabilities and have motivated the rapid development of software tools and execution platforms to design, compile, and run quantum circuits on real hardware [2]. However, most current practical experimentation takes place in the Noisy Intermediate-Scale Quantum (NISQ) regime, where limited qubit quality, noise accumulation, and the absence of full fault tolerance make circuit execution highly sensitive to hardware variability and compilation choices [1,3].

From a data-management perspective, NISQ experimentation poses a specialized repository problem. Experimental outputs are probabilistic rather than deterministic; hardware properties such as coherence times and error rates change over time; and transpilation pipelines introduce intermediate transformations that strongly affect the final executed circuit. As a consequence, the observed outcome of a quantum-circuit run is not sufficiently described by the logical circuit alone. Meaningful interpretation also requires access to execution-time device conditions, calibration snapshots, transpilation decisions, and the links among these artifacts. When such information is omitted or only partially recorded, the resulting datasets become difficult to audit, compare, validate, and reuse.

In this setting, reproducibility cannot be reduced to exact equality of individual outputs. Since quantum measurements estimate probability distributions from finite shots, and since NISQ hardware conditions evolve over time, reproducibility must be interpreted statistically: experimental records should preserve the sampling configuration, calibration state, compilation decisions, and execution context required to compare outcome distributions under documented conditions [2,4–6]. This also requires preserving compiler and workflow context because mapping, routing, and execution orchestration can affect the final circuit and its observed distribution [7–9].

This situation directly impacts reproducibility. Empirical studies have shown that nominally identical quantum experiments can produce different output distributions when executed at different times due to unreported changes in calibration state, coherence behavior, or gate errors [4,5,10]. In many current workflows, critical context—such as initial qubit layout, routing choices, optimization settings, backend descriptors, and calibration freshness—is either lost or preserved only in ad hoc logs. The result is a fragmented experimental record in which the data required for post-mortem analysis, cross-run comparison, and controlled reuse are not managed as first-class repository objects [6,7].

This challenge can be framed as a problem of specialized metadata and schema management rather than only as a problem of quantum software tooling. In domains such as data warehouses, data lakes, and scientific workflows, metadata models, provenance structures, validation contracts, and schema-governance policies are essential to ensure data quality, integrity, interoperability, and traceability [11–14]. The same principles are increasingly necessary in QC, where execution records are contextual, technology-dependent, and temporally unstable. Accordingly, the central need is not merely better logging, but a specialized metadata and schema model capable of supporting governed repositories of quantum-experiment artifacts across heterogeneous backends.

Motivated by this gap, this paper introduces a Quantum Computing–Metadata Model (QC-MM) as a schema-governed metadata model for quantum-circuit experimentation. Its scope is deliberately focused on the experimental record: the logical circuit description, the transpilation trace, the calibration snapshot, the execution context, and the post-run evidence required to interpret probabilistic outcomes. Rather than modeling the full lifecycle of hybrid quantum-classical software, QC-MM focuses on preserving the contextual dependencies that affect traceability and reproducibility in NISQ executions [4,5]. In this sense, the model provides a common structure for representing what was executed, how

it was compiled, under which hardware conditions it was run, and how the resulting evidence can be compared across executions. Such metadata not only improves scientific reporting, but also enables practical downstream benefits. In particular, it supports context- and noise-aware optimization, as well as more principled validation strategies, by making the device- and compilation-level context available as first-class information for analysis and tooling [7,8].

The research question (RQ) guiding this work is the following: How can a specialized metadata and schema model improve the traceability and reproducibility of quantum-circuit experimentation by representing compilation, calibration, and execution context as governed repository data? To answer this RQ, the paper makes three main contributions. First, it defines a lifecycle-oriented metadata model that captures the contextual dependencies of NISQ experimentation in a platform-agnostic form. Second, it formalizes this model as a schema-governed and extensible representation that integrates time-indexed calibration binding, structured transpilation traceability, lightweight provenance links, validation rules, and controlled metadata evolution. Third, it evaluates the proposal through a multi-scenario protocol involving IBM Quantum Cloud, a local SpinQ NMR device, transpilation-focused inspection, repeated-run statistical characterization, and compilation-to-performance traceability. This work considers people working on quantum software and quantum circuit experiments, including practitioners and researchers, those focused on reproducibility and improving quantum programming by leveraging device details, and developers seeking reliable, easier-to-check quantum programs.

The remainder of this paper is organized as follows. Section 2 provides background, related work, a synthesis of technical limitations in the state of the art, and the problem statement. Section 3 describes QC-MM, including its lifecycle model, schema specification, validation rationale, provenance layer, and reference implementation. Section 4 presents the experimental protocol and evaluation scenarios. Section 5 discusses the results, comparative implications, limitations, and future directions. Section 6 concludes the paper.

## 2. Background and Related Work

This section presents background concepts and related work on metadata and reproducibility in quantum-circuit execution. Section 2.1 summarizes the NISQ context, transpilation, and hybrid workflows that make the execution context critical for traceability and reproducibility. Section 2.2 reviews previous research on reproducibility, provenance, and metadata management for quantum circuits and executions. Section 2.3 synthesizes these findings to state the concrete gap addressed by this paper and to justify the need for a dedicated metadata model.

### 2.1. Background

QC departs from classical computation by adopting the qubit as its basic unit of information. While a classical register of $n$ bits represents a single value $x \in \{0, \ldots, 2^n - 1\}$ at any time, an $n$-qubit register is described by a state vector $|\psi\rangle$ in a $2^n$-dimensional Hilbert space, allowing the manipulation of quantum states through superposition, entanglement, and interference [2]. From a computational complexity perspective, this paradigm motivates the class BQP (Bounded-Error Quantum Polynomial Time), which captures decision problems solvable by a quantum computer in polynomial time with bounded error [2]. For specific problem families that exhibit an exploitable mathematical structure, quantum algorithms can yield substantial advantages over classical approaches [2].

In practice, current QC technology operates predominantly in the NISQ era, characterized by devices of intermediate-size without robust fault-tolerant error correction [1]. Consequently, the reliability of circuit execution is tightly constrained by the noise accu-

mulation and the limited coherence time of the physical qubits. Two hardware parameters are particularly relevant for empirical performance: relaxation time ($T_1$), which captures the energy decay from $|1\rangle$ to $|0\rangle$, and dephasing time ($T_2$), which limits how long phase coherence can be preserved [3]. These parameters are not stationary; empirical evidence indicates that they may fluctuate on the hour scale, implying that identical circuits executed on the same device at different times can produce divergent outcome distributions [4,15].

A further practical limitation arises from the restricted connectivity of the device. Contemporary superconducting platforms expose sparse coupling graphs, so two-qubit interactions are only directly available for certain pairs of physical qubits [7]. When a circuit requires entangling qubits that are not physically adjacent, additional routing operations (e.g., SWAP insertions) become necessary. These transformations increase the depth of the circuit and, therefore, the exposure to noise, reducing the final fidelity [7]. As a result, observed performance is not solely a function of the logical circuit, but also of compilation and device-state contingencies at execution time.

To bridge the gap between abstract algorithms and hardware constraints, QC platforms rely on quantum transpilation: a classical pre-processing pipeline that rewrites an ideal circuit into an equivalent circuit executable on a target QPU [8]. Typical stages include (i) decomposition into the native gate set, (ii) selection of the layout (logical mapping to physical qubits), (iii) routing to satisfy connectivity constraints, and (iv) circuit optimization to reduce depth and gate count [8,16,17]. In particular, transpilation decisions are often sensitive to calibration data and error rates; noise-adaptive strategies can significantly affect outcome fidelity [7]. However, many toolchains provide limited transparency about these decisions, complicating later inspection and experimental analysis [9].

Moreover, the most useful applications in the near future will adopt hybrid quantum-classical execution patterns. Variational algorithms such as VQE and QAOA distribute computation across a classical optimizer and a quantum backend: the QPU is used to generate measurement samples, while classical resources update parameters and coordinate iterative execution [1,18]. This iterative workflow introduces bidirectional dependencies between classical control logic and dynamic hardware context (e.g., calibration snapshots and error rates), intensifying the need for systematic capture of execution context [7–9].

In complex data-intensive systems, metadata management provides structured mechanisms to document data properties, provenance, quality, and operational context [19,20]. Established approaches in domains such as data warehouses and data lakes show that formal metadata models support governance, traceability, and process optimization [11,12]. In addition, provenance standards such as the execution history of the W3C-PROV model through entities, activities, and agents enable post-mortem audit and reproducible analysis in distributed scientific workflows [9,13,14]. Translating these principles to NISQ experimentation is particularly relevant because quantum outputs are inherently probabilistic and must be treated as distributions estimated from repeated shots [6]. Consequently, reproducibility in QC is naturally framed as *statistical reproducibility*, where consistent distributions—and the conditions under which they were obtained—matter more than exact value identity [6].

Taken together, the foundational characteristics of QC (probabilistic measurement), NISQ constraints (noise, topology, temporal variability), and transpilation/hybrid workflows (context-sensitive transformations) motivate the need for explicit and structured representations of the compilation and execution context. Such representations provide the conceptual basis for addressing traceability and reproducibility challenges in contemporary quantum software practice.

### 2.2. Related Work

To structure the review, the related work is analyzed according to four gaps identified through a systematic mapping study (SMS) [21] and PRISMA reporting elements [22]. SMS distilled recurring limitations into four gaps (fragmentation/vendor coupling; transpilation opacity; unmanaged calibration volatility; provenance disconnection), and used as design anchors for the QC-MM proposal. These gaps are thoroughly reviewed in [23]. The complete SMS artifacts (protocol, screening log, extraction tables, and primary-study list) are available in [24].

We organize the discussion into four clusters—(i) metadata models and standards, (ii) transpilation and context-aware compilation, (iii) calibration variability and reproducibility, and (iv) provenance and workflow lineage.

#### 2.2.1. Metadata Models and Standards

Metadata is commonly defined as "data about data" and is widely used to support discovery, documentation, interoperability, quality, and traceability in complex information systems [19]. In traditional data engineering, structured metadata underpins governance and lifecycle management in architectures such as data warehouses and data lakes [11,12]. These approaches motivate treating experimental artifacts and results as managed assets rather than ephemeral logs. This idea of managed assets is particularly relevant for hybrid quantum-classical workflows where the same circuit can be iteratively compiled and executed under changing conditions [9].

In the quantum software ecosystem, prior metadata-related proposals include early standardization attempts (e.g., QIS-XML [25]) that focus primarily on logical descriptions and are largely pre-NISQ [26], as well as modeling-oriented approaches (e.g., Quantum UML) that are strong for design-time abstraction but do not directly address runtime execution context and calibration dynamics [27]. System-level perspectives also emphasize that quantum computing services are heterogeneous and rapidly evolving, which increases the importance of portable, platform-agnostic representations [28,29].

#### 2.2.2. Transpilation and Context-Aware Compilation

In the NISQ setting, transpilation/compilation is a critical classical preprocessing step that maps an abstract circuit to hardware constraints (basis gates, connectivity, routing) and can substantially alter circuit depth and error accumulation [7,8,16]. Retargetable compilation toolchains (e.g., $t|ket\rangle$) highlight cross-backend compilation as a practical need, but compilation decisions often remain internal to the toolchain, reducing transparency for post-mortem analysis [8].

A central line of work in NISQ compilation uses device noise and topology to guide mapping and optimization [7,30]. However, even when calibration-aware strategies exist, the traceability of compilation decisions (e.g., selected initial layout or routing decisions) is frequently not exported as structured machine-interpretable metadata, which complicates inspection and controlled comparisons between runs and optimization settings [7,31].

#### 2.2.3. Calibration Variability and Reproducibility

A key obstacle to reproducibility in NISQ experiments is temporal variability in hardware behavior (e.g., coherence times, readout/gate error rates), which can induce run-to-run differences even under nominally identical conditions [4,10,15]. This empirical variability motivates a shift from "exact reproducibility" to statistical reproducibility, where results must be interpreted as distributions and compared with explicit documentation of the execution context [6,32].

Consequently, reproducible reporting requires the capture of not only the logical circuit and the final compiled circuit, but also time-indexed calibration information (snapshots, validity windows, and freshness) that can explain the drift of the results and support fair cross-run comparisons [5,6].

#### 2.2.4. Provenance and Workflow Lineage

Provenance, or data lineage, provides formal descriptions of the entities, activities, and agents involved in the production and transformation of data [13]. The W3C PROV data model offers a domain-agnostic standard to represent provenance structures and relations [14]. In quantum workflows, lifecycle and provenance perspectives have been proposed to make hybrid executions traceable and expose "black box" behavior in quantum toolchains [9,33]. More recent work also explores the analysis and provenance of integrated workflows in quantum contexts and the orchestration of middleware at the level of heterogeneous offerings [34,35].

Nevertheless, a persistent challenge is operational adoption: provenance approaches can remain too heavyweight or too detached from everyday SDK usage, creating friction for routine experimental scripting. This motivates lightweight provenance layers that can be embedded in practical workflows while maintaining the ability to reconstruct the flow: *design* → *compile* → *execute* the lineage [33].

Among the reviewed approaches, QProv is the closest to QC-MM because it explicitly addresses provenance collection for quantum computations [33]. However, QC-MM differs in its primary focus: rather than proposing a provenance system, it defines a schema-governed metadata model for experiment records, with explicit support for calibration validity semantics, repeated-run execution contexts, schema validation, and controlled metadata evolution.

To make the positioning of QC-MM more explicit, Table 1 expands the comparison beyond conceptual coverage and includes operational criteria relevant to repository-oriented NISQ experimentation. The goal is not to rank previous approaches, but to clarify which concerns are addressed as first-class metadata-management capabilities. In particular, the comparison highlights whether each approach explicitly supports runtime execution context, calibration freshness semantics, pass-level transpilation metadata, backend integration, repository readiness, and scalability considerations.

As shown in Table 1, existing approaches provide important but partial support for the requirements of traceable NISQ experimentation. Early circuit-description standards and design-time modeling approaches are useful for representing logical artifacts, but do not explicitly address runtime calibration, execution context, or repository-level validation. Provenance- and workflow-oriented proposals provide stronger lineage support, but they are often defined at a higher level of abstraction and do not necessarily preserve calibration snapshots, transpilation decisions, and execution outcomes within a single schema-governed experiment record. Calibration-aware compilation tools, in turn, can use hardware information internally, but their decisions are typically not persisted as standardized queryable metadata linked to the final execution result.

QC-MM differs from these approaches by combining the relevant dimensions in a single repository-oriented metadata model. It explicitly binds time-indexed calibration snapshots to execution records, captures transpilation information as structured metadata, supports repeated-run statistical reporting, includes lightweight provenance links, and formalizes the resulting record through a JSON Schema contract with versioning and controlled extensibility. This positioning clarifies the intended contribution of QC-MM: it is not a replacement for provenance standards or compiler optimizers but a schema-governed

metadata layer that makes quantum-circuit experiment records validable, comparable, evolvable, and reusable across heterogeneous backends.

**Table 1.** Comparative positioning of QC-MM with respect to representative metadata, provenance, workflow, and compilation-oriented approaches.

| Criterion | QIS-XML | Quantum UML | QProv | Workflow/ Lifecycle Approaches | Calibration-Aware Compilation Tools | QC-MM |
|---|---|---|---|---|---|---|
| Logical circuit representation | Explicit | Explicit | Partial | Partial | Input-level | Explicit |
| Runtime execution context | No | No | Partial | Partial | Partial | Explicit |
| Time-indexed calibration binding | No | No | Partial | Partial | Used internally | Explicit |
| Calibration validity/freshness semantics | No | No | No/Partial | Partial | Partial | Explicit |
| Structured transpilation trace | No | No | Partial | Partial | Tool-dependent | Explicit |
| Pass-/decision-level compilation metadata | No | No | Partial | Partial | Usually internal | Explicit |
| Repeated-run/statistical support | No | No | Partial | Partial | No/Partial | Explicit |
| Lightweight provenance linkage | No | No | Explicit | Explicit | No | Explicit |
| Schema-based validation | Partial | No | No/Partial | No/Partial | No | Explicit |
| Controlled versioning/extensibility | No | No | No/Partial | Partial | Tool-dependent | Explicit |
| Backend/platform agnosticism | Partial | Design-level | Partial | Partial | Usually backend-dependent | Explicit |
| Practical SDK/backend integration | Legacy/ limited | Modeling-level | Requires integration | Middleware-oriented | Tool-native | Adapter-based |
| Repository-ready experiment record | No | No | Partial | Partial | No | Explicit |
| Storage/scalability strategy | Not addressed | Not addressed | Not central | Not central | Not central | Initial; extensible |

*Note:* "Explicit" means that the capability is addressed as a central feature of the approach; "Partial" means that it is addressed indirectly, at a different abstraction level, or only for a subset of the concern; "No" means that the capability is not central to the approach; "Tool-dependent" indicates that support depends on a specific implementation. Representative references: QIS-XML [25], Quantum UML [27], QProv [33], workflow/lifecycle approaches [34,35], and calibration-aware compilation tools [7,13,30,31].

#### 2.2.5. Synthesis of Thecnical Limitations in the State of the Art

The reviewed literature shows that the limitations of current approaches are not isolated, but structural. Early metadata standards and modeling approaches provide useful abstractions for representing logical circuits or design-time information, but they do not capture the runtime evidence required in NISQ experimentation, such as time-varying calibration snapshots, backend-specific error indicators, or the compiler decisions that transform a logical circuit into a hardware-executable artifact [25–27]. Conversely, calibration-aware and noise-adaptive compilation techniques use device information internally to improve mapping or routing decisions, but they usually do not expose these decisions as persistent, machine-interpretable metadata linked to the corresponding execution outcomes [7,13,30,31].

A second limitation concerns provenance and workflow lineage. Provenance-oriented approaches provide a valuable conceptual foundation to reconstruct how experimental artifacts are produced and transformed [11,12,33–35]. However, they are often defined at the workflow or service-orchestration level and may remain detached from the lightweight scripting practices commonly used in quantum-circuit experimentation. As a result, they do not necessarily provide a repository-ready record that simultaneously binds the logical circuit, the transpilation trace, the calibration state, the execution context, and the probabilistic outcome distribution [4–6,8].

Finally, existing approaches tend to address only part of the reproducibility problem. Some models emphasize interoperability, others provenance, others calibration, and others compilation optimization. Nevertheless, NISQ reproducibility requires these dimensions

to be represented together. Without an explicit schema-governed structure, experimental records remain difficult to validate, compare, evolve, and reuse across heterogeneous backends. This motivates the need for a specialized metadata model that treats quantum-circuit executions as governed repository artifacts rather than as isolated logs or backend-specific output files.

#### 2.2.6. Positioning and Technical Contribution of QC-MM

QC-MM is positioned as a repository-oriented metadata and schema model for NISQ quantum-circuit experimentation. Its contribution is not limited to defining a list of metadata fields; rather, it integrates several mechanisms that are usually treated separately in prior work. First, QC-MM structures the experimental record according to the design–compilation–execution–analysis lifecycle, preserving explicit links between logical intent, compiler transformations, backend context, and post-run evidence [17,33]. Second, it introduces time-indexed calibration binding, where calibration snapshots are represented as validity-scoped artifacts and linked to concrete execution records [5,6,8]. Third, it makes transpilation inspectable by storing compiler configuration, optimization levels, pass-level information, and pre/post-compilation metrics as structured metadata [7,13].

In addition, QC-MM incorporates a lightweight provenance layer inspired by standard provenance concepts, but adapted to low-overhead experimental scripting [11,12]. This layer records identifiers and relations among the main artifacts without requiring a full workflow-provenance infrastructure. Finally, QC-MM formalizes the resulting model through a JSON Schema contract, enabling validation before persistence, controlled schema evolution through versioning, and extensibility for backend-specific attributes [36]. In this sense, QC-MM complements existing standards, provenance models, and compiler-oriented approaches by providing a unified, validation-ready, and evolvable metadata record for traceable quantum-circuit experiments across heterogeneous backends.

### *2.3. Problem Statement*

Despite rapid progress in quantum software toolchains, current experimental practice still lacks a specialized and machine-interpretable data model for managing quantum-circuit execution records as governed repository artifacts. In particular, existing workflows do not systematically preserve the interplay between transpilation decisions, time-varying device calibration, and run outputs in a form that supports validation, traceability, controlled reuse, and long-term interoperability. As a consequence, experimental records are often reduced to partial logs or isolated outputs, rather than being managed as structured data assets suitable for repository-oriented analysis. This limitation directly undermines reproducibility in NISQ experimentation and reveals a broader metadata-management gap in an emerging computational domain.

From the perspective of specialized data management, this gap is not only a matter of missing contextual information but also of missing schema governance, lifecycle-aware metadata management, and traceable repository design. In addition, provenance considers the documentation of entities, processes, and agents that determine the origin and transformation of data [13], and standards such as W3C-PROV operationalize this notion through entities, activities, and agents [14]. In large-scale data architectures, provenance is a cornerstone of data quality and post-mortem analysis [9,12]. Quantum-circuit experimentation produces heterogeneous and strongly contextual data: logical circuit descriptions, compilation traces, backend descriptors, calibration snapshots, and probabilistic results. These artifacts evolve over time, depend on platform-specific constraints, and must remain explicitly linked if the resulting datasets are to be interpretable and reusable. Without a formal metadata and schema layer, such artifacts cannot be consistently validated, com-

pared between executions, or integrated into repositories that support data quality and post-mortem analysis [33].

The problem is further aggravated by the temporal instability of the NISQ hardware. The properties of the devices fluctuate over short time windows, and the quality of an execution may depend on the precise calibration state and the compiler configuration in effect at run time [4]. Recent work advocates *statistical reproducibility*, where the consistency of output distributions must be evaluated together with complete traceability of the execution environment [6]. If outputs are stored without explicit calibration binding, structured transpilation traceability, and provenance-aware cross-references, the resulting records become difficult to interpret retrospectively and nearly impossible to compare under controlled conditions. Thus, the issue is not merely incomplete reporting, but the absence of a specialized metadata infrastructure capable of representing contextual and time-varying execution data with sufficient integrity and granularity.

Accordingly, the main problem addressed in this paper is the lack of a platform-agnostic metadata and schema model that can support traceable quantum-experiment repositories by (i) representing the full circuit lifecycle context, (ii) binding time-indexed calibration and compilation information to execution outcomes, and (iii) enabling validation, controlled evolution, and lightweight provenance across heterogeneous backends. In response, this paper positions QC-MM as a specialized data model for repository-oriented management of quantum-circuit experimentation, rather than as an ad hoc logging mechanism. Table 2 summarizes the gaps (G1–G4) that motivate this proposal.

**Table 2.** Traceability gaps (G), identified problem and evidence/rationale.

| G | Identified Problem | Evidence/Rationale |
|---|---|---|
| **G1** | Fragmentation and lack of agnosticism | Heterogeneous quantum services and rapidly evolving offerings motivate portable representations [28,29]; early standards and design-time models do not fully address NISQ runtime context [26,27]. |
| **G2** | Opacity in transpilation | Compilation/mapping decisions can strongly affect depth and outcomes [7,30]; retargetable compilation highlights cross-backend compilation, traceability is often insufficient [8]; compiler strategy selection remains non-trivial [31]. |
| **G3** | Unmanaged volatility | Temporal variability in hardware and performance drift challenge reproducibility [4,5]; statistical reproducibility requires context capture [6]. |
| **G4** | Provenance disconnection | Provenance foundations and standards exist [13,14]; QC lifecycle/provenance proposals motivate lineage capture but require practical integration [9,33]. |

Based on the four identified gaps (G1–G4), we derive six design requirements (R1–R6) that define the capabilities needed to address the problem described above.

- **R1 Interoperability / platform agnosticism.** The model must support heterogeneous quantum platforms and technology families, reducing vendor lock-in and fragmentation [26,28,29].
- **R2 Unified lifecycle coverage;** The model must coherently represent the circuit lifecycle (design–compilation-execution-analysis) to enable consistent reporting and comparison across runs and studies [9].
- **R3 Structured transpilation traceability.** The model must record compilation choices (e.g., optimization strategy, mapping/routing decisions) as machine-interpretable metadata to mitigate transpilation opacity [7,8,31].

- **R4 Time-indexed calibration binding.** The model must capture time-stamped calibration snapshots (and their validity/freshness semantics) and bind them to executions to support statistical reproducibility [4–6].
- **R5 Lightweight provenance linkage.** The model must include a practical provenance layer aligned with standard models to reconstruct design → compile → execute lineage with low overhead [13,14,33].
- **R6 Evolvability and governance.** The model must support controlled evolution (versioning/extensibility) and validation to remain usable under rapidly changing platforms and metadata needs [9,11,12].

To address these gaps, we propose QC-MM as a specialized metadata and schema model for traceable quantum-experiment repositories, designed to support validation, interoperability, provenance-aware linkage, and controlled metadata evolution across heterogeneous quantum platforms.

## 3. QC-MM: Proposal and Specification

This section presents the QC-MM proposal and is organized as follows. Section 3.1 summarizes the model and its rationale for designing it. Section 3.2 presents the metadata lifecycle and its core entities and relationships. Section 3.3 details the JSON Schema specification and validation rules, including versioning and extensibility. Section 3.4 introduces a lightweight provenance and traceability layer for identifiers, linking, and traceability between artifacts. Finally, Section 3.5 outlines the reference implementation and its integration with representative quantum software platforms.

### *3.1. Overview and Design Rationale*

QC-MM is specified as a structured metadata and schema model for quantum-circuit experiment records across the compilation, calibration, execution, and post-run analysis stages in the NISQ era. Its purpose is to make explicit the contextual evidence that relates four elements: the logical circuit descriptor, the compilation trace, the calibration snapshot associated with the backend state, and the execution record containing run outputs. The conceptual structure was represented in UML to clarify entities and relationships, and the resulting specification was formalized as a JSON Schema [36] to support validation and interoperable serialization. This scope avoids modeling the full hybrid quantum-classical software lifecycle and instead focuses on the information required to interpret, compare, and reproduce NISQ experiment executions. At the requirements level, QC-MM addresses R1–R6 by separating static device descriptors from time-varying calibration snapshots, recording compilation decisions and their effects, binding compilation, calibration, and execution results within a single experiment record, and adding lightweight provenance-oriented links for lineage reconstruction. Table 3 provides a requirement-to-mechanism traceability map to make the rationale of the design traceable.

### *3.2. Metadata Lifecycle Model (Design–Compilation–Execution–Analysis)*

QC-MM structures metadata according to a lifecycle that mirrors the end-to-end experimental workflow: *design*, *compilation*, *execution*, and *analysis*. The lifecycle is represented through a set of core entities and their relations using a UML class diagram. Figure 1 shows how QC-MM organizes and links information between classical and quantum layers, preserving explicit relationships among circuit definition, compilation, calibration, execution, and analysis records.

#### 3.2.1. Design (Logical Intent)

The entity `CircuitMetadata` captures the pre-compilation description of the circuit, including descriptive attributes (e.g., algorithm family, authorship), structural metrics (e.g., logical depth and number of logical qubits), and the original circuit source (e.g., QASM).

#### Compilation (Transformations and Decisions)

The CompilationTrace entity `CompilationTrace` records transpilation decisions and their measurable effects. It stores, for example, the optimization level and the sequence of compilation passes, and it quantifies transformations by comparing pre- and post-compilation circuit characteristics (e.g., depth changes). This supports a later analysis of compilation choices and their implications.

**Table 3.** Traceability from requirements (R) to QC-MM entities/fields and enforcement mechanisms.

| R | QC-MM Entity/Field(s) | Mechanism |
|---|---|---|
| **R1** | *QCMetadataModel*; *DeviceMetadata* & *CalibrationData* separation; user-defined properties | Platform-agnostic JSON representation & JSON Schema contract; vendor-specific attributes captured via a controlled extension area while keeping a stable validated core. |
| **R2** | *CircuitMetadata*, *CompilationTrace*, *ExecutionContext* | Lifecycle structuring with explicit relations and identifiers; execution_context modeled as 1:N to support repeated runs under a uniform context. |
| **R3** | *CompilationTrace*: optimization level; compilation_passes; pre/-post structural metrics | Structured compilation trace captured at pass-level + measurable deltas to inspect compiler decisions; schema constraints ensure presence and consistent typing of trace fields. |
| **R4** | *CalibrationData*: timestamp_captured; valid_until; qubit_level indicators; *ExecutionContext* binds compilation & calibration & outputs | Time-indexing in UTC using ISO-8601 across lifecycle entities; explicit validity windows and binding through ExecutionContext links; schema-required fields enforce non-missing temporal metadata. |
| **R5** | *ProvenanceRecordLean*: provenance_id, timestamp_recorded; relations linking artifacts | Lightweight provenance layer grounded in PROV concepts but implemented as a minimal DAG of identifiers; relations provide traceable lineage. |
| **R6** | *QCMetadataModel.model_version*; schema version fields; deprecations policy; user-defined properties | Explicit versioning to support controlled evolution; backward-compatible changes handled via additive fields and extensions; breaking changes restricted to major bumps; deprecated fields remain supported for a defined transition window. |

#### Execution (Device Context and Results)

To characterize the target device, QC-MM separates `DeviceMetadata`, relatively static properties (e.g., topology and native gate set) from `CalibrationData`, which captures time-version snapshots of device state, including qubit-level coherence indicators (e.g., $T_1$ and $T_2$) and gate fidelity metrics. Execution outcomes are modeled through `ExecutionContext`, which binds a specific compilation instance and a calibration snapshot to the corresponding run outputs (e.g., counts and derived probabilities). The model supports multiple executions in the same context, enabling a consistent grouping of multiple shot batches.

#### Temporal and Validity Semantics (Calibration Binding)

A core design choice in QC-MM is to represent calibration as a *time-indexed* and *validity-scoped* artifact. Each `CalibrationData` instance includes (i) `timestamp_captured`, which records when the calibration snapshot was obtained, and (ii) `valid_until`, which defines the intended validity window of that snapshot. This explicit temporal semantics enables

calibration-to-execution binding, which is essential for interpreting NISQ results under device variability and to support statistical reproducibility across runs [4–6]. QC-MM operationalizes this by linking each execution record to the calibration snapshot that was in effect at execution time, making the hardware state traceable rather than implicit [33].

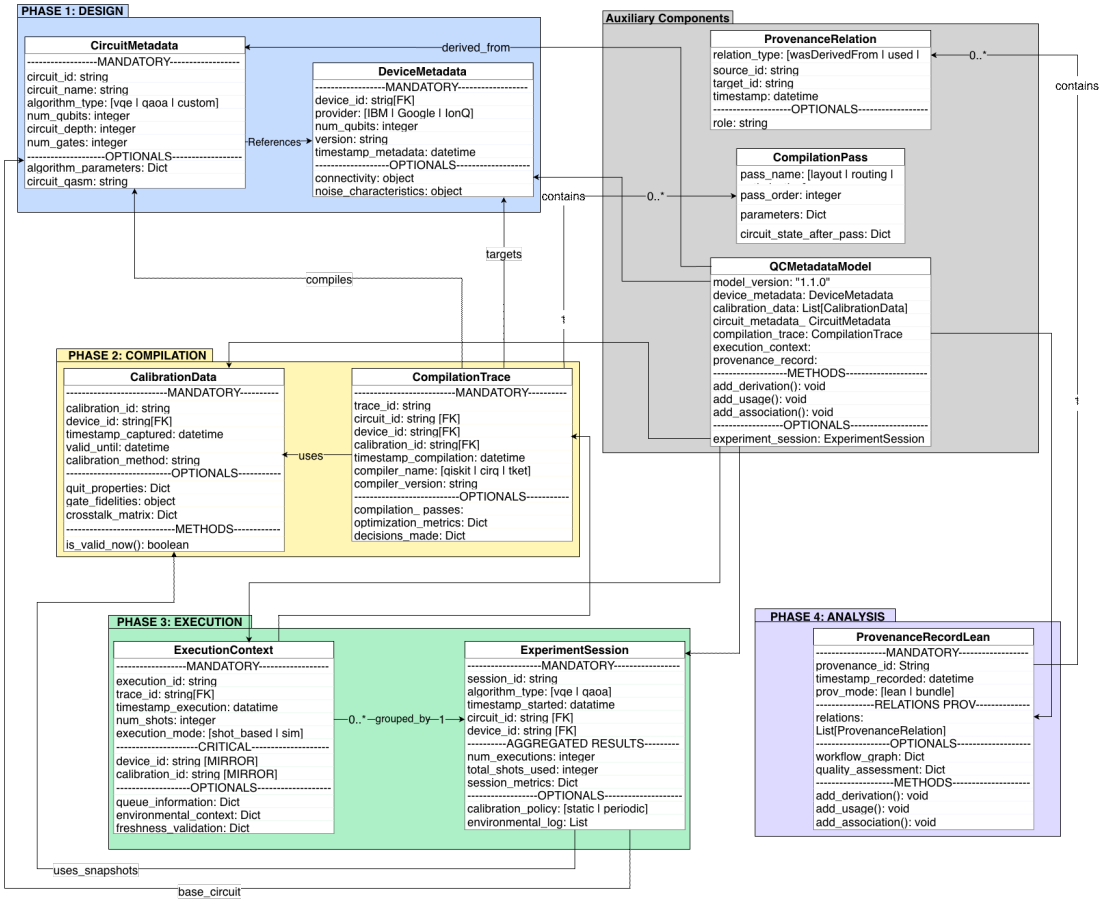


**Figure 1.** Full view of the QC-MM conceptual model.

One-to-Many Execution Records (Replications)

QC-MM models `execution_context` as a 1:N relation: one instance of `QCMetadataModel` may contain a list of `ExecutionContext` records, each corresponding to a concrete run (or batch) of the same experimental intent under potentially distinct compilation choices, calibration windows, or repetition settings. This design supports *replicated runs* and longitudinal experimentation as first-class citizens, enabling controlled comparisons while preserving the full context per-run required for inspectability and reproducibility [6,37].

Analysis (Post-Run Interpretation)

Although QC-MM does not prescribe a single analysis method, the lifecycle structure allows downstream analyzes to be grounded on explicit contextual variables (e.g., compilation configuration and calibration state), facilitating controlled comparisons and reproducibility-oriented reporting.

### *3.3. Schema Specification and Validation*

To maximize interoperability, QC-MM is materialized as a technical specification using JSON for serialization and JSON Schema for formal constraints. We adopt JSON Schema Draft-07, enabling explicit integrity rules such as mandatory fields (`required`), conditional dependencies (`if-then-else`), and format constraints via regular expressions [36].

Temporal consistency is supported by systematically incorporating timestamps in UTC across lifecycle entities, using ISO 8601 to ensure an unambiguous and universally interpretable event timeline. Listing 1 shows a concrete implementation of these rules for the calibration module.

**Listing 1.** Excerpt of a JSON schema for calibration validation.

```
    "calibration_data": {
      "type": "array",
      "minItems": 1,
      "items": {
        "type": "object",
        "required": [
          "calibration_id",
          "device_id",
          "timestamp_captured",
          "valid_until",
          "calibration_method",
          "calibration_version"
        ],
        "properties": {
          "calibration_id": {"type": "string"},
          "device_id": {"type": "string"},
          "timestamp_captured": {"type": "string", "format": "date-time"},
          "valid_until": {"type": "string", "format": "date-time"},
          "calibration_method": {"type": "string"},
          "calibration_version": {"type": "string"},
          "qubit_properties": {"type": "object"},
          "gate_fidelities": {"type": "object"},
          "crosstalk_matrix": {"type": "object"},
          "additional_metrics": {"type": "object"}
        }
      }
    }
```

In addition to syntactic constraints, QC-MM validates metadata during generation and ingestion, before persistence, by checking each metadata instance against the JSON Schema specification. This prevents non-conformant records from entering the repository and supports consistent downstream processing. This validation follows a *schema-based validation strategy*, where metadata objects are checked against the formal specification, reducing the risk of corrupted or non-conformant instances entering the repository.

#### 3.3.1. Rationale for JSON Schema

JSON Schema Draft-07 was selected because it provides a practical balance between interoperability, formal validation, tooling maturity, and operational simplicity [36]. JSON is a lightweight, text-based and language-independent format for structured data interchange, which makes it suitable for repository records that must be inspected, exchanged, and processed across heterogeneous software stacks. This is especially relevant in quantum-computing workflows, where experiment metadata may be produced by Python 3.9-based scripts, cloud APIs, local provider adapters, validation tools, and downstream data-management systems.

JSON Schema complements this representation by providing a declarative contract for the structure and constraints of JSON documents [38]. In QC-MM, this contract is used to enforce required fields, data types, temporal formats, object cardinalities, and controlled extension points before metadata records are persisted. Draft-07 was adopted because it is stable, widely supported by validation libraries, and expressive enough for the requirements of the model, including required, format, pattern, object composition, arrays, and conditional constraints [39]. Compared with XML Schema, JSON Schema reduces integration friction with modern API- and SDK-based quantum workflows. Compared with ontology-oriented representations such as RDF/OWL, it provides less semantic ex-

pressiveness but a lower operational burden for routine experimental scripting. Compared with binary schema formats such as Protocol Buffers or Avro, it favors human inspectability and repository transparency, while still allowing future compact encodings to be derived from the same canonical metadata structure.

#### 3.3.2. Validation Rules for Temporal Fields and Execution Cardinality

To make the temporal context traceable, QC-MM enforces explicit timestamp formats and validity constraints at the schema level. All temporal attributes (e.g., timestamp_captured, `valid_until`, and execution timestamps) are encoded as ISO-8601 `date-time` strings and validated through the schema contract. In addition, QC-MM introduces a consistency rule for calibration validity: `valid_until` must not precede `timestamp_captured`. This constraint prevents ill-formed calibration bindings and supports reliable post-mortem analysis under device drift [4–6].

QC-MM also validates execution recording as a first-class mechanism for reproducibility. The root container requires `execution_context` to be a list (1:N) of `ExecutionContext` objects, allowing multiple replications and longitudinal runs to be stored under a single experiment record. At minimum, each `ExecutionContext` must bind (i) the compiled circuit reference, (ii) the calibration snapshot reference (when applicable), and (iii) the run outputs, ensuring that each execution is interpretable as a traceable unit [6].

#### 3.3.3. Versioning and Extensibility

Given the rapid evolution of NISQ platforms and the emergence of new calibration and execution indicators, QC-MM is designed for extension without destabilizing its core. Following the Open/Closed Principle [40], the model adopts a *hybrid schema* strategy: a rigid *core* section preserves stable, validated metadata, while a complementary *user-defined properties* section supports future or experiment-specific attributes as generic key–value pairs. This approach follows established strategies for metadata management on heterogeneous and evolving sources [12] and preserves backward compatibility without requiring schema refactoring whenever vendors introduce new parameters.

This policy is operationalized through semantic versioning in the root object (`model_version`) and an additive-first evolution rule, with explicit deprecation windows to break transitions.

#### 3.3.4. Versioning and Backward Compatibility

QC-MM treats the schema as a long-lived contract that must evolve without disrupting existing repositories and downstream tooling. We adopt a strict semantic versioning policy for both the schema and the root metadata object: `model_version = MAJOR.MINOR.PATCH`. MAJOR increments indicate breaking changes (e.g., remove/rename required fields or changes in meaning). MINOR increments indicate backward-compatible extensions (e.g., new optional fields or new entity variants that do not invalidate existing instances). PATCH increments indicate clarifications or constraint refinements preserving the instance validity.

To preserve backward compatibility under rapid platform evolution, QC-MM follows an "additive-first" rule: new platform indicators are introduced as optional fields in the core only when stable, otherwise they are captured through the user-defined properties section (hybrid schema). When a breaking redesign becomes unavoidable, fields are first marked as `deprecated` for at least one MINOR cycle, while remaining accepted by validators; producers should emit both the legacy and successor fields during the transition. Validators remain lenient for deprecated fields within the transition window, while consumers can implement explicit migrations keyed on `model_version`.

### 3.4. Provenance and Traceability Layer

To provide end-to-end traceability, QC-MM integrates a lightweight provenance layer, `ProvenanceRecordLean`, grounded in the W3C-PROV conceptualization of lineage as a directed acyclic graph that links entities, activities, and agents [13,14]. The goal is to capture a minimal yet sufficient trace: stable identifiers for key lifecycle artifacts, explicit links between design/compilation/execution records, and audit-friendly references that enable reconstruction of *what was executed*, *how it was compiled*, and *under which device conditions*. This provenance layer supports post-mortem analysis, accountability, and reproducibility-oriented reporting with low overhead.

### 3.5. Reference Implementation

A reference prototype was implemented to demonstrate technical feasibility and operationalize QC-MM in realistic workflows. This prototype comprises four main components: (i) metadata classes implementing the QC-MM entities, (ii) provider adapters for obtaining backend- and execution-specific information, (iii) a schema validator that checks generated records against the JSON Schema specification, and (iv) an export/persistence interface that stores conformant metadata records. Unlike a provenance collector focused primarily on event capture, the prototype generates complete QC-MM instances that combine circuit, compilation, calibration, execution, and provenance-oriented links, collectively under a common schema.

The implementation follows an object-oriented design and incorporates established design patterns to preserve maintainability and extensibility [41,42]. In particular, to handle heterogeneity across quantum backends, the prototype employs the Adapter pattern [42], enabling QC-MM to interact with diverse execution environments (e.g., vendor platforms and local simulators) through a uniform interface without coupling to provider-specific APIs. As part of quality assurance, the implementation integrates metadata validation during generation and ingestion (e.g., via `jsonschema`), ensuring that generated metadata instances satisfy the formal specification before being stored in the final repository (e.g., a data-lake-style persistence layer). This enforces consistent, analysis-ready metadata and supports reliable downstream processing.

## 4. Results

This section presents the evaluation of QC-MM in representative quantum-software workflows. Section 4.1 describes the common experimental protocol and measurement procedure. The subsequent Sections 4.2–4.6 report five scenarios: cloud-based execution on IBM Quantum, local interoperability on a SpinQ NMR device, transpilation-effect inspection, repeated-run statistical characterization, and compilation-to-performance traceability. Detailed experimental configurations, extended tables, and additional supporting figures are provided in the Appendix A.

### 4.1. Experimental Protocol and Measurement Procedure

The evaluation was designed to assess whether QC-MM can generate valid, traceable, and comparable metadata records across representative quantum-experiment workflows. All scenarios followed a common metadata-generation pipeline: logical circuit definition, backend selection, transpilation, calibration capture when available, execution, QC-MM record generation, JSON Schema validation, and persistence of the resulting metadata artifact. The workloads included VQE, QFT, and Grover circuits because they exercise complementary aspects of the model: variational execution, structural compilation effects, and repeated probabilistic sampling.

The independent variables considered in the evaluation were the target backend, algorithm family, and transpiler optimization level. The dependent variables were circuit depth, CNOT count, compilation time, execution counts, derived probability distribution, schema-validation result, and an estimated fidelity proxy. The fidelity proxy was computed from the normalized measurement counts as the probability mass assigned to the expected or target output state. This fidelity proxy is not intended to replace full quantum-state fidelity; rather, it provides a lightweight execution-level indicator for comparing runs under the same workload and measurement basis. A QC-MM record was considered valid when it satisfied the JSON Schema contract and preserved the required cross-references among the circuit metadata, compilation trace, calibration snapshot, execution context, and provenance record.

### *4.2. Scenario A: Capturing Dynamic Hardware Context on IBM Quantum Cloud*

Scenario A evaluates the feasibility of using QC-MM to capture dynamic hardware context in an IBM Quantum Cloud setting. Specifically, this scenario applies the execution-record and calibration-snapshot entities to bind time-varying backend conditions to probabilistic outcomes through explicit provenance links. The objective is to assess whether QC-MM can preserve execution-time device information as traceable metadata in a real cloud-based workflow.

We executed a VQE workload on the superconducting backend `ibm_fez` and recorded metadata throughout the full cloud execution path. A key observation is that QC-MM separates design-time artifacts from execution-time context: the logical circuit is defined client-side, while calibration metadata is acquired and linked only once the job reaches physical execution after queueing and runtime orchestration.

The generated QC-MM instance includes a time-indexed calibration snapshot bound to the execution record, with qubit-level indicators (e.g., $T_1$, $T_2$, and readout error) captured at the moment of execution. This makes hardware heterogeneity explicit and traceable *a posteriori*—for example, QC-MM reveals substantial differences in coherence properties between the qubits involved in the run, enabling a principled interpretation of outcome variability beyond what traditional execution logs provide. As shown in Figure 2, the VQE experiment was not deployed directly on quantum hardware; instead, it was executed through the public cloud infrastructure. The numbered steps in the figure clarify the operational sequence of this workflow: the logical circuit is first defined in the client environment and submitted through the `IBMProvider` adapter; the job is then managed by `Qiskit Runtime REST API`, including queueing and orchestration before physical execution on the target QPU. Once the execution is completed, QC-MM captures the corresponding calibration snapshot and execution results, links them to the compiled circuit and provenance record, validates the generated metadata against the JSON Schema contract, and persists the resulting QC-MM record as a repository-ready artifact. Listing 2 shows an excerpt from an execution record showing calibration metadata captured from the IBM backend at run time.

### *4.3. Scenario B: Technology Interoperability on a Local SpinQ NMR Device*

Scenario B evaluates the interoperability of QC-MM across heterogeneous quantum technologies in a local SpinQ Gemini Pro (2 qubits) NMR setting. Using the same lifecycle entities and schema contract, this scenario assesses whether QC-MM can generate valid and structurally consistent metadata in an on-premise execution environment without modifying the underlying JSON representation. The objective is to determine whether the model remains portable across different hardware and communication architectures. Unlike the cloud scenario, this on-premise workflow relies on a low-latency synchronous communication channel (TCP/IP socket) and does not depend on internet connectivity.

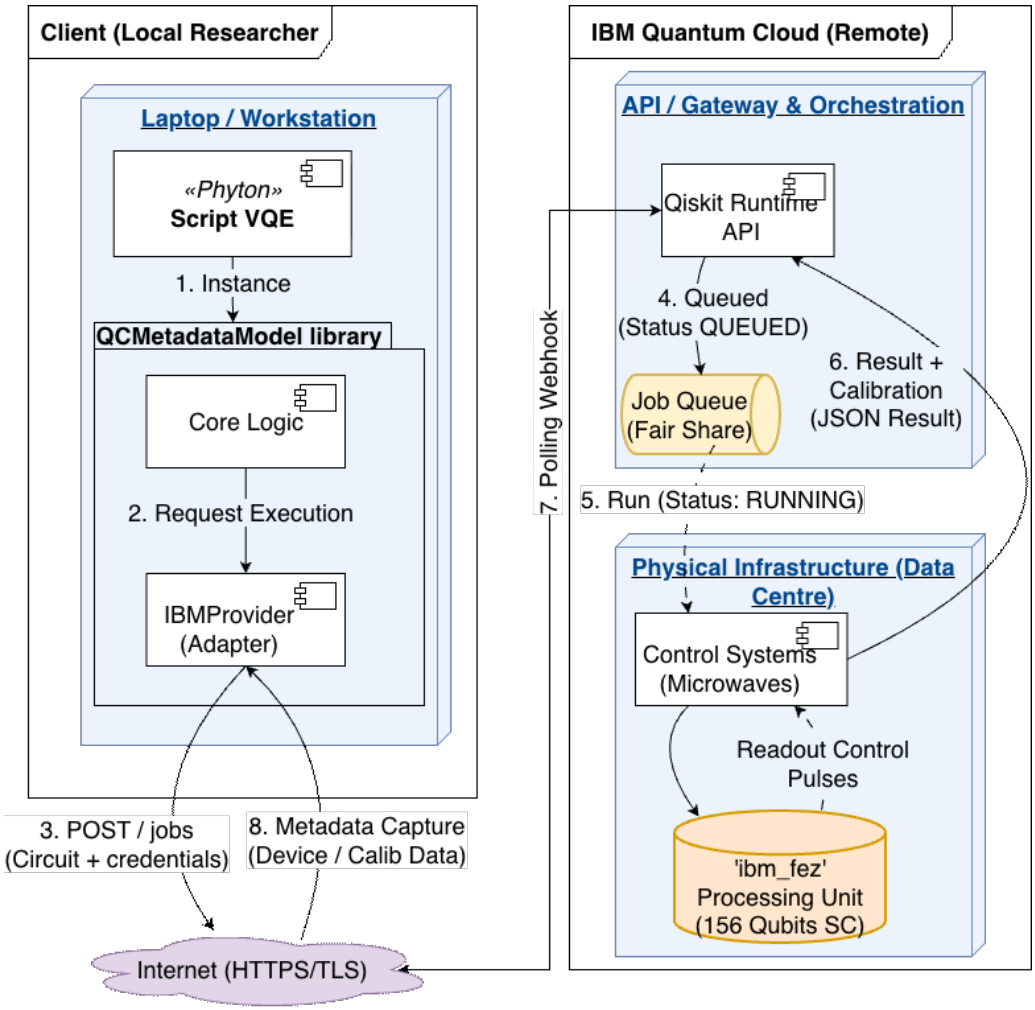


**Figure 2.** IBM Quantum Cloud workflow for QC-MM metadata generation. Numbered steps indicate the sequence from logical circuit definition to runtime orchestration, QPU execution, calibration capture, QC-MM record generation, schema validation, and metadata persistence.

**Listing 2.** Excerpt of a JSON execution record-calibration metadata from the IBM backend.

```
"calibration_data": [
  {
    "calibration_id": "cal_ibm_fez_20251119_161617",
    "device_id": "ibm_fez",
    "timestamp_captured": "2025-11-19T15:36:11-03:00Z",
    "valid_until": "2025-11-20T15:36:11-03:00Z",
    "calibration_method": "ibm_quantum_api",
    "calibration_version": "1.0",
    "qubit_properties": {
      "0": {
        "t1_us": 61.03977241917603,
        "t2_us": 53.7863580439973,
        "readout_error": 0.009521484375
      },
      "1": {
        "t1_us": 206.0749258658043,
        "t2_us": 269.87469385888727,
        "readout_error": 0.0225830078125
      },
```

Using a provider adapter, the workflow produced a valid QC-MM metadata file without modifying the JSON structure. Device characteristics specific to NMR (e.g., native all-to-all connectivity between nuclei) were populated in the backend descriptor while preserving the same schema constraints. This scenario supports the claim that QC-MM decouples logical experimental intent from hardware-specific realization, enabling consistent reporting across disparate technologies (superconducting cloud vs. local NMR). Figure 3 shows the local workflow used to operate the SpinQ Gemini NMR device. Unlike the cloud-based scenario, this execution does not involve asynchronous queueing or remote runtime orchestration; instead, the client workstation communicates with the device through a synchronous TCP/IP connection. The numbered steps in the figure clarify the operational sequence: local circuit definition, invocation of the SpinQ provider adapter, opening of the TCP/IP communication, translation of the abstract circuit into device-specific commands, execution on the SpinQ NMR device, capture of device and execution metadata, QC-MM record generation, JSON Schema validation, and metadata persistence.

To illustrate how QC-MM supports on-premise execution through heterogeneous transport mechanisms, Listing 3 presents the implementation of the `SpinQBackend` adapter. This component extends the base provider class and acts as the translation layer between the abstract metadata model and the physical NMR device.

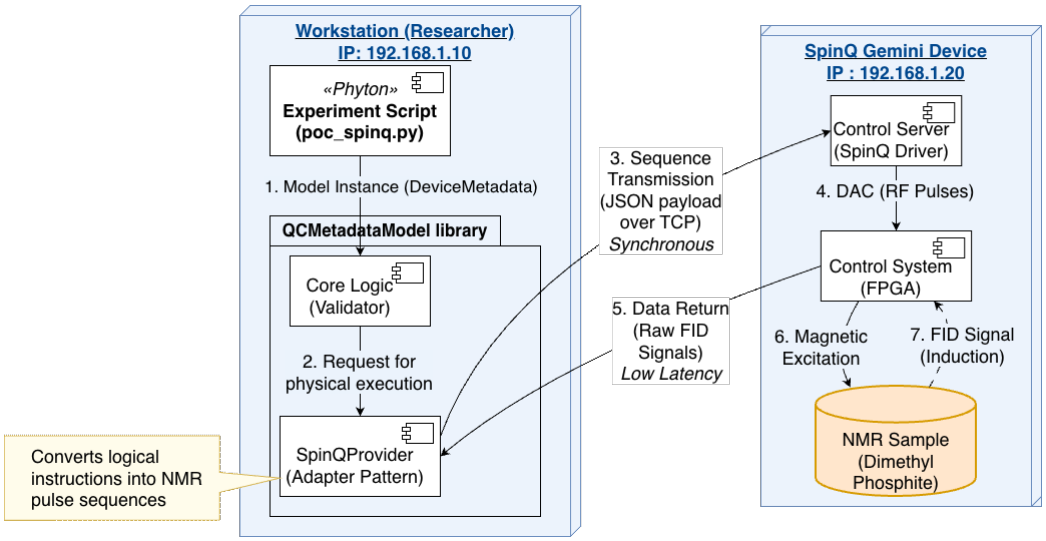


**Figure 3.** Local SpinQ NMR workflow for QC-MM metadata generation. Numbered steps indicate local circuit definition, SpinQ adapter invocation, TCP/IP communication, device-specific translation, NMR execution, metadata capture, QC-MM record generation, schema validation, and persistence.

The inclusion of this code fragment is essential to demonstrate the flexibility of the transport layer in the proposed model. In contrast to the IBM adapter, which manages the execution of asynchronous jobs through HTTP-based cloud services, the implementation in Listing 3 relies on a synchronous block socket connection. This difference shows that the metadata model is not coupled to a specific data-transmission protocol. In addition, the script captures the native hardware configuration of the SpinQ device—namely, its all-to-all topology between the phosphorus and hydrogen nuclei—and injects it into the `DeviceMetadata` entity without requiring any modification to the JSON schema structure. The successful execution of `poc_spinq.py` using this adapter produced a valid metadata file, confirming that QC-MM decouples the logical intent of the experiment from the physical complexity of the underlying hardware and can support heterogeneous technologies, such as superconducting and NMR platforms, under a common reporting standard.

**Listing 3.** Excerpt of the class SpinQProvider (Python coding).

```
class SpinQProvider:
    """Wrapper for SpinQ NMR Quantum Computer"""

def __init__(self, ip: str, port: int = 8989,
             username: str = None, password: str = None):
        """
        Initialize SpinQ provider

        Args:
            ip: IP address for quantum computer
            port: communication port (default: 8989)
            username: user authentication
            password: password authentication
        """
        self.ip = ip
        self.port = port
        self.username = username
        self.password = password
        self.engine = None
        self.compiler = None
        self._initialize_service()
```

### *4.4. Scenario C: Verifying Optimization Effects Through Transpilation Records*

This scenario evaluates whether QC-MM can make transpilation effects traceable through structured compilation records. Using a 4-qubit QFT circuit compiled under different Qiskit optimization levels, this experiment assesses whether the model can capture pass-level decisions and quantify their structural impact on the compiled circuit. The objective is to determine whether QC-MM supports instance-level inspection of optimization effects rather than treating transpilation as an opaque transformation.

We selected a 4-qubit QFT circuit as workload due to its sensitivity to connectivity constraints and compiled it multiple times using Qiskit transpilation optimization levels.

The resulting transpilation record captured pass-level information and explicit optimization metrics. The metadata indicates that the most aggressive configuration reduced the CNOT count by 40% relative to the baseline, and it also records the activation of additional heuristic passes (e.g., swap mapping) that explain differences in compilation time. These results show that QC-MM makes transpilation outcomes inspectable at the instance level, enabling debugging-focused comparisons between compilation configurations.

### *4.5. Scenario D: Statistical Validation Across Algorithm Families*

Scenario D evaluates the ability of QC-MM to support repeated-run statistical characterization across different algorithm families. Running Grover, QFT, and VQE workloads in a common metadata structure, this experiment evaluates whether the model can consistently capture execution results, replications, and variability indicators such as confidence intervals. The objective is to determine whether QC-MM provides a suitable basis for statistical reproducibility in the NISQ experimentation.

Beyond the VQE workload, we also run Grover and QFT circuits on IBM Quantum hardware, each with five independent repetitions, allowing 95% confidence intervals for a fidelity proxy based on measured ground-state probability from the execution record.

QC-MM successfully captured comparable performance metadata across structurally different algorithms. For example, Grover (2 qubits) exhibited a tight 95% CI of $\pm 0.0010$ around a mean fidelity proxy of 0.9610, while QFT with 3 qubits showed a wider CI of $\pm 0.0188$ (mean 0.9218), reflecting higher observed variability. These results indicate that

QC-MM supports the statistical characterization of NISQ variability while maintaining a consistent metadata structure between workloads.

### 4.6. Scenario E: Tracing Compilation Decisions to Execution Performance

This scenario evaluates whether QC-MM can preserve explicit traceability between compilation decisions and execution-level performance. Starting from the same baseline circuit and varying the level of optimization of the transpiler, this experiment assesses whether the model can link compilation metadata—such as circuit depth and compilation time—to execution results in a structured and traceable manner. The objective is to determine whether QC-MM supports reproducibility-oriented analysis of how software-toolchain choices affect observed performance.

Starting from the same baseline circuit, we generated four transpiled variants by varying the compiler optimization level from 0 to 3. QC-MM recorded, for each variant, the original and compiled circuit depth, compilation time, and the associated fidelity proxy from execution, thereby enabling a direct link between compilation metadata and results.

The recorded traces show that higher optimization levels substantially reduce circuit depth (from 7 to 1 in the reported runs) while simultaneously improving the fidelity proxy (from 0.90 at baseline to 0.96 at maximum optimization). Importantly, QC-MM preserves these relationships as structured, queryable evidence through cross-references between the transpilation record and execution record. This supports reproducibility-oriented analysis by making it possible to attribute performance changes to concrete compilation decisions rather than treating the toolchain as an opaque transformation pipeline. Additional details on the implementation and measurement-level details are reported in the Appendix A to facilitate independent validation and replication.

## 5. Discussion

The implementation and evaluation of QC-MM make it possible to discuss the proposal not only as a contribution to quantum software experimentation but also as a contribution to specialized metadata management in an emerging computational domain. From this perspective, the central problem addressed by QC-MM is the lack of a governed, machine-interpretable, and evolvable representation for storing and relating quantum-experiment artifacts as repository data.

### 5.1. Relevance of QC-MM as a Specialized Metadata and Schema Model

QC-MM is particularly relevant as a response to ecosystem fragmentation. By combining a UML-based conceptual representation with a JSON Schema specification, QC-MM provides both a clear information structure and a validation-ready serialization format. The proposal moves beyond earlier efforts that focused primarily on static circuit descriptions (e.g., QIS-XML) and lacked explicit support for dynamic execution context. In QC-MM, the key novelty is that circuit artifacts are not treated in isolation: compilation decisions, device state, and results are modeled as first-class linked information.

From the perspective of specialized data management, QC-MM contributes in at least four ways: (i) it supports data modeling for repository-oriented representation; (ii) strengthens data governance through validation rules and controlled structure; (iii) preserves contextual integrity through time-indexed calibration binding; and (iv) enables controlled metadata evolution through semantic versioning and extensibility mechanisms.

From a software quality standpoint, QC-MM improves two properties that are critical for empirical quantum research. First, *portability* is reinforced by separating the logical description of an experiment from the physical backend context, mitigating provider lock-in, and facilitating reuse across heterogeneous platforms. Second, *analyzability* is strengthened

by explicitly recording transpilation decisions, turning the transpiler from an opaque "black box" into a traceable process. Unlike optimization-centric approaches that transform circuits without exposing standardized traces, QC-MM provides structured evidence of mapping and routing decisions, enabling investigators to distinguish logical-design issues from compilation-induced artifacts.

Finally, QC-MM incorporates a lightweight provenance layer that links probabilistic outcomes to the calibration state captured at a specific time instant. This pragmatic provenance design enables data lineage and post-mortem analysis without the overhead of full workflow provenance systems, and it directly targets the volatility of NISQ hardware by preserving calibration as part of the experimental evidence.

### 5.2. Interpretation of the Evaluation Results from a Data-Management Perspective

The evaluation results suggest that structured metadata management is not merely an auxiliary documentation layer but a necessary condition for reliable management of quantum-experiment data.

First, the portability results indicate that a single metadata representation can describe executions across different technologies, supporting the claim that QC-MM operates as an abstraction layer over heterogeneous backends. This is a relevant claim given the recurring concerns about limited standardization and the risk of vendor-specific coupling.

Second, the benchmarking results highlight the value of QC-MM for debugging and optimization evaluation. By quantifying compilation effects (e.g., reductions in depth associated with aggressive optimization levels), QC-MM provides instance-level explainability of transpilation outcomes. Prior compiler work reports efficiency improvements, but typically does not offer a standardized, user-facing mechanism to verify those improvements on a per-experiment basis. In contrast, QC-MM enables traceable comparisons between pre- and post-transpilation artifacts and supports systematic reporting.

Third, integration with cloud-based execution demonstrates the feasibility of capturing dynamic hardware indicators (e.g., $T_1$, $T_2$) without disrupting the workflow. This supports the broader thesis that calibration-aware approaches are important in NISQ settings, and it extends that idea by emphasizing calibration preservation as traceable evidence for explaining fidelity degradation that would be invisible in conventional execution logs.

### 5.3. Comparison with Related Approaches and Implications for Specialized Repository Design

QC-MM advances the state of practice by integrating, in a single open and implementable specification, (i) time-indexed calibration snapshots, (ii) a standardized transpilation trace, and (iii) a provenance-oriented linkage among artifacts and results. In contrast to proposals that remain largely conceptual, QC-MM is accompanied by a reference implementation that can be integrated into real toolchains, lowering adoption barriers and enabling reproducibility-oriented workflows in practice.

From an adoption perspective, a JSON-based specification is compatible with object storage and CI/CD-like pipelines, making it suitable for continuous experimentation and regression detection. Moreover, explicit schema versioning supports sustainability by enabling backward-compatible evolution as platforms expose new backend parameters or new technology families become relevant.

Its contribution lies not in proposing a general-purpose data model but in defining the structured, governed, and evolvable metadata layer required for trustworthy repository management in quantum experimentation.

### 5.4. Threats to Validity

Several limitations must be acknowledged. Regarding *external validity*, the experiments were necessarily limited in circuit size due to access limitations and focused on

representative workloads; a broader evaluation across larger devices and a wider range of circuit families remains required. However, incorporating multiple algorithmic families reduces the risk that the conclusions are specific to a single class of circuits.

For *internal validity*, the lack of control over the state of cloud hardware is an inherent challenge. However, QC-MM partly reframes this limitation as an observable variable: rather than assuming stable conditions, the model captures and documents device-state heterogeneity, reducing the risk of incorrectly attributing hardware-induced effects to logical circuit design or software faults.

A further concern is *scalability*. Rich metadata may lead to growth in storage volume as circuit sizes and compilation graphs expand. At larger scales, plain-text JSON may become inefficient for high-throughput campaigns or large compilation graphs. A practical evolution path is to preserve JSON Schema as the canonical validation contract while allowing compact physical encodings such as CBOR, or BSON-like binary document representations, for storage and transmission [43,44]. In addition, provenance-heavy deployments could persist QC-MM identifiers and relations in graph databases, enabling efficient traversal of design–compile–execute lineages across large experiment repositories [45].

QC-MM supports *traceability* through explicit links, but it does not by itself provide tamper-evident storage or cryptographic integrity guarantees; these mechanisms are left for future repository-level extensions.

### 5.5. Open Challenges and Future Directions for Metadata-Governed Repositories

Beyond the findings reported in Section 4, QC-MM also points to clear open challenges and future research directions. First, structured calibration metadata can enable JIT-assisted compilation, where logical-to-physical mappings and routing choices are adapted using near-real-time stability indicators such as T1, T2, readout error, and gate fidelity [46]. This direction aligns with just-in-time transpilation, noise-aware compilation, and time-varying processor models [5,7], while QC-MM would preserve the selected mapping and calibration identifier as traceable metadata. Second, scalability remains a practical concern as circuit sizes, compilation graphs, and experimental throughput increase. A possible evolution path is to preserve JSON Schema as the canonical validation contract while exploring compact encodings such as CBOR or BSON for storage and transmission, and graph databases for efficient traversal of provenance-heavy design–compile–execute lineages [43–45]. Third, ecosystem alignment suggests an additional line of work: the maturity of the specification makes QC-MM a plausible candidate for standardization-oriented integration, for instance, as an extension layer compatible with OpenQASM and other intermediate representations used in QC toolchains [47].

A related open challenge concerns iterative hybrid quantum-classical workflows. Although the evaluation in this paper uses VQE, QFT, and Grover primarily as benchmark circuit executions, QC-MM is structurally compatible with algorithms in which a classical optimizer or controller repeatedly generates, compiles, executes, and updates quantum circuits, such as QAOA-assisted Benders decomposition and adaptive Grover-based search procedures [48,49]. In such cases, each iteration can be represented as a distinct ExecutionContext linked to the corresponding CircuitMetadata, CompilationTrace, CalibrationData, and run outputs, while ProvenanceRecordLean can preserve the iteration-to-iteration lineage among classical parameters, generated circuits, measurement results, and subsequent update decisions. For example, a QAOA- or adaptive-Grover-based hybrid workflow could be stored as a sequence of metadata records sharing a common experiment identifier but differing in iteration index, parameter vector, compiled circuit, calibration snapshot, and measured distribution. This conceptual walkthrough indicates that QC-MM does not require a different schema for hybrid loops; rather, hybrid execution can be represented

through repeated, provenance-linked execution contexts, with future work focusing on richer optimizer-state metadata and large-scale validation on iterative workloads.

Crucially, these directions remain within the scope of the article and clarify the contribution to the RQ. QC-MM directly addresses the objective of proposing a metadata model for managing quantum-circuit artifacts across the *design–compilation–execution–analysis* lifecycle by providing (i) an explicit separation of logical artifacts and backend context, (ii) a standardized record of transpilation decisions, and (iii) time-indexed calibration snapshots bound to execution outcomes. The results show that making compilation, execution, and calibration contexts explicit and machine-interpretable is a key enabler of traceability and reproducibility in NISQ experimentation, as it supports traceable lineage, instance-level explainability of transpilation effects, and interpretation of probabilistic outcomes under changing hardware conditions.

Overall, QC-MM contributes an initial but operational foundation for treating quantum experimentation as a metadata-management problem grounded in traceability, validation, and controlled schema evolution.

## 6. Conclusions

This work addressed the reproducibility challenge of NISQ quantum-circuit experimentation by proposing QC-MM, a schema-governed metadata model for traceable experiment records. The model represents and relates logical circuit descriptors, compilation traces, calibration snapshots, execution records, and post-run evidence through a JSON Schema specification that supports validation, portability, and controlled metadata evolution across heterogeneous backends.

The evaluation demonstrates the feasibility and usefulness of QC-MM in practice. A reference prototype captures execution and compilation context with sufficient granularity to improve interpretability and enable metadata-informed compilation, while a lightweight provenance component binds outputs to the calibration state at a specific time, supporting end-to-end traceability; in the reported experiments, this translated into an 85.7% reduction in circuit depth and a 6.7% improvement in estimated fidelity proxy.

The main contribution of QC-MM is an operational foundation for traceability-by-design in quantum experimentation, complementing improved reporting with machine-actionable context for analysis. Nevertheless, the current evidence is bounded by the evaluated backends and circuit sizes, and broader validation on larger devices and more diverse workloads remains necessary.

Future work will extend QC-MM to contexts such as JIT compilation, where freshness-aware metadata can guide near real-time optimization. We will explore alignment with ecosystem standards (e.g., OpenQASM) as well as potential contributions to standardization efforts. We aim to consolidate QC-MM as a practical step toward reproducible and traceable quantum experimentation grounded in software and data engineering principles.

**Author Contributions:** Conceptualization, N.H. and S.S.; methodology, S.S.; software, N.H.; validation, S.S., A.F. and N.H.; formal analysis, S.S. and A.F.; investigation, N.H.; resources, S.S.; data curation, N.H.; writing—original draft preparation, N.H. and S.S.; writing—review and editing, S.S. and A.F.; visualization, S.S. and N.H.; supervision, S.S.; project administration, S.S.; funding acquisition, S.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** Samuel Sepúlveda is funded by ANID–Chile, Fondecyt Iniciación, grant No. 11240702.

**Data Availability Statement:** The complete SMS artifacts (protocol, screening log, extraction tables, and primary-study list) are available as an external resource [24]. Data and software artifacts supporting the findings of this study are publicly available at the GitHub repository: https://github.com/nawels693/model_meta_data, accessed on 20 January 2026. This repository contains the

Python implementation of the qc-metadata-model library, the corresponding JSON Schemas used for validation, the proof-of-concept scripts for IBM Quantum and SpinQ, and the raw data generated in the benchmarking experiments.

**Acknowledgments:** Thanks to RIPAISC-Red Iberoamericana Para el Avance de la Ing. de Software Cuántico (CyTED 525RT0174). Samuel and Nawel would also like to thank Ania Cravero and Fernanda Gutiérrez for their useful technical support. During the preparation of this manuscript/study, the authors used ChatGPT 5.2 and Grammarly to assist with English language, including rewriting sentences, restructuring paragraphs to improve clarity and readability, and editing LaTeX. These tools were not used to generate scientific ideas, interpret results, or produce new scientific content. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| API | Application Programming Interface |
| BQP | Bounded-Error Quantum Polynomial Time |
| CI/CD | Continuous Integration/Continuous Deployment |
| ISO | International Organization for Standardization |
| JSON | JavaScript Object Notation |
| NISQ | Noisy Intermediate-Scale Quantum |
| NMR | Nuclear Magnetic Resonance |
| OpenQASM | Open Quantum Assembler |
| PRISMA | Preferred Reporting Items for Systematic reviews and Meta-Analyses |
| QAOA | Quantum Approximate Optimization Algorithm |
| QASM | Quantum Assembler |
| QC | Quantum Computing |
| QC-MM | Quantum Circuit Metadata Model |
| QFT | Quantum Fourier Transform |
| QIS-XML | Quantum Information Science-Extensible Markup Language |
| QPU | Quantum Processor Unit |
| RQ | Research Question |
| SMS | Systematic Mapping Study |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| UML | Unified Modeling Language |
| VQE | Variational Quantum Eigensolver |
| W3C-PROV | World Wide Web Consortium Provenance |

## Appendix A. QC-MM: Detailed Specification and Experimental Evidence

This appendix provides (i) a detailed account of the QC-MM information architecture and its lifecycle modules, (ii) implementation-level mechanisms used to enforce data integrity and traceability (schema-based validation), and (iii) a structured description of proof-of-concept scenarios and quantitative results. The intent is to preserve continuity in the paper while making supporting technical details available in a self-contained artifact.

### *Appendix A.1. Prototype Implementation and Enforcement Mechanisms*

#### Appendix A.1.1. Reference Implementation and Lifecycle Aggregation

QC-MM is implemented as a Python reference library that supports controlled aggregation of lifecycle artifacts. In particular, the root container `QCMetadataModel` enforces `execution_context` as a list (1:N), enabling repeated runs and longitudinal experimentation to be represented under a single experiment record. Safe insertion is supported through

methods such as `add_execution_context`, avoiding duplicate references and supporting traceable record growth.

#### Appendix A.1.2. JSON Schema Contract and Validation

Beyond in-memory representation, QC-MM is formalized as a JSON Schema contract to enable language-agnostic validation of metadata instances before downstream processing. QC-MM highlights enforcement mechanisms in the calibration module as representative examples.

- **Cardinality constraints**: e.g., `minItems: 1` to prevent empty calibration lists.
- **Mandatory fields**: `required` keys enforce the presence of identifiers and validity fields (including `valid_until`).
- **Temporal standardization**: ISO-8601 `date-time` formats enforce unambiguous temporal indexing for `timestamp_captured` and `valid_until`.

These controls operationalize calibration-to-execution binding under hardware volatility and support post-mortem verification. This is in line with empirical evidence that device characteristics change over time and can alter observed performance [4–6].

### *Appendix A.2. Scenarios and Detailed Results*

This section reports the details of the quantitative results used in the main paper, including additional context and interpretation, too detailed for the manuscript body.

#### Appendix A.2.1. Scenario C: Verification Optimization Effects Through Transpilation Records

**Objective.** Validate whether `CompilationTrace` captures transpilation decisions and enables post-mortem analysis of compilation strategies.

**Method.** A 4-qubit QFT circuit is transpiled four times under preset optimization levels (0–3). The goal is not to benchmark the vendor compiler but to verify that QC-MM persistently records structural deltas between strategies.

**Key observations.** The recorded metadata evidences a drastic reduction in circuit depth at higher optimization levels, and captures pass-level evidence and optimization metrics that enable inspection beyond the final results [7,8].

**Table A1.** Compilation metrics automatically captured by the model.

| OpLv | OrD | FinD | EstF | CoT (ms) |
|---|---|---|---|---|
| 0 | 7 | 7 | 0.90 | 745.12 |
| 1 | 7 | 7 | 0.92 | 75.12 |
| 2 | 7 | 1 | 0.94 | 80.82 |
| 3 | 7 | 1 | 0.96 | 76.72 |

*OpLv: Optimization Level, OrD: Original Depth, FinD: Final Depth, CoT: Compilation Time.*

#### Appendix A.2.2. Details of Scenario D: Statistical Validation Across Algorithm Families

**Objective.** Validate QC-MM under diverse workloads (Grover, QFT, and VQE reference) on an IBM Quantum backend, using $N = 5$ independent repetitions per circuit (where applicable) to compute 95% confidence intervals and assess stochastic variability. The experimental design follows common empirical software engineering guidance on repeatability and reporting context [37].

**Table A2.** Results of Scenario D, statistical validation with algorithm diversity.

| Benchmark | Qubits | N | Mean Fidelity | 95% CI |
|---|---|---|---|---|
| Grover (Search) | 2 | 5 | 0.9610 | ±0.0010 |
| QFT (Basis Change) | 3 | 5 | 0.9218 | ±0.0188 |
| QFT (Basis Change) | 4 | 5 | 0.9104 | ±0.0079 |
| VQE ($H_2$ reference) | 2 | 1 | 0.4482 | N/A |

The wider CI observed for QFT_3q (±0.0188) indicates run-to-run fluctuations that would likely be missed under single-run reporting, reinforcing the need for execution-context binding and replication-aware data structures. This aligns with the larger call for statistical reproducibility under NISQ variability [6].

Appendix A.2.3. Deatils of Scenario E: Tracing Compilation Decisions to Execution Performance

**Objective.** Evaluate whether QC-MM can trace how compilation choices affect the final circuit topology and execution-level performance by correlating `CompilationTrace` with `ExecutionContext`.

**Method.** A base circuit is transpiled (optimization levels 0–3). QC-MM records depth changes and compilation time, linking them to execution performance indicators.

**Table A3.** Results of Scenario E: traceability of optimization and performance.

| OpLv | OrD | CoD | CoT (ms) | EstF * |
|---|---|---|---|---|
| 0 (baseline) | 7 | 7 | 745.12 | 0.90 |
| 1 | 7 | 7 | 75.12 | 0.92 |
| 2 | 7 | 1 | 80.82 | 0.94 |
| 3 (max) | 7 | 1 | 76.72 | 0.96 |

*OpLv: Optimization Level, OrD: Original Depth, CoD: Compiled Depth, CoT: Compilation Time, EstF: Estimated Fidelity.* * *Extracted from the correlation analysis between* `CompilationTrace` *and* `ExecutionContext`.

## References


1. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
2. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*; Cambridge University Press: Cambridge, UK, 2010.
3. Krantz, P.; Kjeldsen, M.; Lin, F.; Bylander, J.; Oliver, W.D. A quantum engineer's guide to superconducting qubits. *Appl. Phys. Rev.* **2019**, *6*, 021318.
4. Dasgupta, S.; Humble, T.S. Characterizing the reproducibility of noisy quantum circuits. *Entropy* **2022**, *24*, 244. [CrossRef] [PubMed]
5. Etxezarreta Martinez, J.; Fuentes, P.; deMarti iOlius, A.; Garcia-Frias, J.; Fonollosa, J.R.; Crespo, P.M. Multiqubit time-varying quantum channels for NISQ-era superconducting quantum processors. *Phys. Rev. Res.* **2023**, *5*, 033055.
6. Senapati, P.; Wang, Z.; Jiang, W.; Humble, T.S.; Fang, B.; Xu, S.; Guan, Q. Towards redefining the reproducibility in quantum computing: A data analysis approach on nisq devices. In *Proceedings of the 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), Bellevue, WA, USA, 17–22 September 2023*; IEEE: Piscataway, NJ, USA, 2023; Volume 1, pp. 468–474.
7. Murali, P.; Linke, N.M.; Martonosi, M.; Abhari, A.J.; Nguyen, N.H.; Adler, C.H. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Providence, RI, USA, 13–17 April 2019; pp. 1015–1029.
8. Sivarajah, S.; Dilkes, S.; Cowtan, A.; Simmons, W.; Edgington, A.; Duncan, R. t|ket>: A retargetable compiler for NISQ devices. *Quantum Sci. Technol.* **2020**, *6*, 014003. [CrossRef]
9. Weder, B.; Barzen, J.; Leymann, F.; Vietz, D. The Quantum Software Lifecycle. In Proceedings of the 1st ACM SIGSOFT International Workshop on Quantum Software Engineering (Q-SE), Online, 9 November 2020; pp. 2–9.

10. Tannu, S.S.; Qureshi, M.K. Not all qubits are created equal: A case for variability-aware policies for NISQ grids. In Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Providence, RI, USA, 13–17 April 2019; pp. 987–999.
11. Inmon, W.H. *Building the Data Warehouse*, 4th ed.; John Wiley & Sons: Hoboken, NJ, USA, 2005.
12. Sawadogo, P.; Darmont, J. On data lake architectures and metadata management. *J. Parallel Distrib. Comput.* **2021**, *152*, 128–140.
13. Buneman, P.; Khanna, S.; Tan, W.C. Why and Wherefore of Provenance. In *Proceedings of the 8th International Conference on Database Theory (ICDT), London, UK, 4–6 January 2001*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 316–330.
14. Moreau, L.; Missier, P.; Cheney, J.; Soiland-Reyes, S. *PROV-N: The Provenance Notation*; World Wide Web Consortium: Wakefield, MA, USA, 2013. Available online: http://www.w3.org/TR/prov-n/ (accessed on 11 June 2026).
15. Carroll, M.; Rosenblatt, S.; Jurcevic, P.; Lauer, I.; Kandala, A. Dynamics of superconducting qubit relaxation times. *npj Quantum Inf.* **2022**, *8*, 132. [CrossRef]
16. Li, G.; Ding, Y.; Xie, Y. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. In Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Providence, RI, USA, 13–17 April 2019; pp. 1001–1014.
17. Duncan, R.; Kissinger, A.; Perdrix, S.; Van De Wetering, J. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Quantum* **2020**, *4*, 279. [CrossRef]
18. McClean, J.R.; Romero, J.; Babbush, R.; Aspuru-Guzik, A. The theory of variational hybrid quantum-classical algorithms. *New J. Phys.* **2016**, *18*, 023023. [CrossRef]
19. Gilliland-Swetland, A.J. *Setting the Stage*, 3rd ed.; Getty Research Institute: Los Angeles, CA, USA, 2008.
20. Khan, A.; Ahmad, A.; Waseem, M.; Liang, P.; Fahmideh, M.; Mikkonen, T. Software Architectures for Quantum Computing Systems: A Systematic Review. *J. Syst. Softw.* **2023**, *201*, 111682. [CrossRef]
21. Petersen, K.; Vakkalanka, S.; Kuzniarz, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Inf. Softw. Technol.* **2015**, *64*, 1–18. [CrossRef]
22. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ* **2021**, *372*, n71. [CrossRef] [PubMed]
23. Sepúlveda, S.; Huenchuleo, N. Metadata models for Quantum Circuits Experiments: A Systematic Mapping. In Proceedings of the Argentinean Symposium on Quantum Computing (ASQC 2026), La Plata, Argentina, 10–13 August 2026.
24. Sepúlveda, S.; Huenchuleo, N. Metadata Models for Hybrid Quantum-Classical Systems: A Systematic Mapping Study Protocol. 2026. Available online: https://doi.org/10.17605/OSF.IO/C3SDH (accessed on 11 June 2026). [CrossRef]
25. Heus, P.; Gomez, R. Qis-xml: An extensible markup language for quantum information science. *arXiv* **2011**, arXiv:1106.2684.
26. Heus, J.; Gomez, P. Towards a standardized description of quantum algorithms. In Proceedings of the International Conference on Quantum Information, Rochester, NY, USA, 13–15 June 2007.
27. Pérez-Castillo, R.; Piattini, M. Design of classical-quantum systems with UML. *Computing* **2022**, *104*, 2375–2403. [CrossRef]
28. Yang, Z.; Zolanvari, M.; Jain, R. A survey of important issues in quantum computing and communications. *IEEE Commun. Surv. Tutor.* **2022**, *25*, 1059–1094. [CrossRef]
29. LaRose, R. Overview and comparison of gate level quantum software platforms. *Quantum* **2019**, *3*, 130. [CrossRef]
30. Murali, P.; McKay, D.; Martonosi, M.; Javadi-Abhari, A. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Lausanne, Switzerland, 16–20 March 2020.
31. Quetschlich, N.; Burgholzer, L.; Wille, R. MQT Predictor:Automatic device selection with device-specific circuit compilation for quantum computing. *ACM Trans. Quantum Comput.* **2025**, *6*, 1–26. [CrossRef]
32. National Academies of Sciences, Engineering, and Medicine. *Reproducibility and Replicability in Science*; The National Academies Press: Washington, DC, USA, 2019. [CrossRef] [PubMed]
33. Weder, B.; Barzen, J.; Leymann, F.; Salm, M.; Wild, K. QProv: A provenance system for quantum computing. *IET Quantum Commun.* **2021**, *2*, 171–181. [CrossRef]
34. Weder, B.; Barzen, J.; Beisel, M.; Leymann, F. Provenance-Preserving Analysis and Rewrite of Quantum Workflows for Hybrid Quantum Algorithms. *SN Comput. Sci.* **2023**, *4*, 233. [CrossRef]
35. Weder, B.; Barzen, J.; Beisel, M.; Bühler, F.; Georg, D.; Leymann, F.; Stiliadou, L. Qunicorn: A middleware for the unified execution across heterogeneous quantum cloud offerings. In *Proceedings of the 2025 IEEE/ACM International Workshop on Quantum Software Engineering (Q-SE), Ottawa, ON, Canada, 3 May 2025*; IEEE: Piscataway, NJ, USA, 2025; pp. 17–24.
36. Wright, A.; Andrews, H.; Hutton, B.; Dennis, G. JSON Schema Validation: A Vocabulary for Structural Validation of JSON. 2018. Draft-07. Available online: https://json-schema.org/draft-07/json-schema-validation (accessed on 10 May 2026).
37. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2012.

38. JSON Schema Organization. JSON Schema Specification. 2024. Available online: https://json-schema.org/specification (accessed on 11 June 2026).
39. Bray, T. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, Internet Engineering Task Force. 2017. Available online: https://doi.org/10.17487/RFC8259 (accessed on 2 June 2026). [CrossRef]
40. Martin, R.C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*; Prentice Hall: Hoboken, NJ, USA, 2018.
41. Booch, G.; Maksimchuk, R.A.; Engle, M.W.; Young, B.J.; Conallen, J.; Houston, K.A. *Object-Oriented Analysis and Design with Applications*, 3rd ed.; Addison-Wesley Professional: Boston, MA, USA, 2007.
42. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley: Boston, MA, USA, 1994.
43. Bormann, C.; Hoffman, P. Concise Binary Object Representation (CBOR). RFC 8949. 2020. Available online: https://doi.org/10.17487/RFC8949 (accessed on 11 June 2026). [CrossRef]
44. BSON. BSON Specification Version 1.1. 2024. Available online: https://bsonspec.org/spec.html (accessed on 15 June 2026).
45. Angles, R.; Gutierrez, C. Survey of Graph Database Models. *ACM Comput. Surv.* **2008**, *40*, 1–39. [CrossRef]
46. Wilson, E.; Singh, S.; Mueller, F. Just-in-time quantum circuit transpilation reduces noise. In *Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), Denver, CO, USA, 12–16 October 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 345–355.
47. Cross, A.W.; Bishop, L.S.; Smolin, J.A.; Gambetta, J.M. Open quantum assembly language. *arXiv* **2017**, arXiv:1707.03429.
48. Zhao, Z.; Fan, L.; Guo, Y.; Wang, Y.; Han, Z.; Hanzo, L. QAOA-Assisted Benders' Decomposition for Mixed-Integer Linear Programming. In *Proceedings of the ICC 2024–IEEE International Conference on Communications, Denver, CO, USA, 9–13 June 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 1127–1132.
49. Zheng, X.; Lin, D.; Chen, H. Grover Adaptive Search-Based Hybrid Benders Decomposition for Mixed-Integer Linear Programs. *IEEE Trans. Quantum Eng.* **2026**, *7*, 1–23. [CrossRef]