

# QDSV Bridge: Problem-First Semantic-to-OpenQASM Workflows for Reproducible Quantum Software Engineering

Jaime Alexander Jimenez Lozano  
QDSV / Qruba, Colombia  
jaimeajl@qruba.site

## ABSTRACT

Quantum software workflows often move rapidly from a domain problem to circuit construction, requiring users to choose encodings, gates, ansatz structures, or backend-specific implementation details before the problem intent and constraints are explicitly represented. This paper presents QDSV Bridge, an open-source Python SDK that supports problem-first semantic specifications and exports inspectable OpenQASM/Qiskit-oriented artifacts, reproducibility reports, and expert-oriented construction inputs. Rather than replacing circuit frameworks, Bridge acts as an upstream workflow layer: users declare a bounded problem family, state-space structure, signals, goals, limits, and target artifact format; Bridge then validates the specification and returns problem-derived artifacts with traceability metadata, warnings, digests, and reproducibility evidence. The contribution is positioned as a quantum software engineering interface for traceable semantic-to-circuit handoff. We describe the public developer-preview architecture, supported delivery modes, artifact contract, Qiskit/OpenQASM workflow, and current boundaries. Bridge is listed as a Qiskit Ecosystem project and is available through PyPI, public documentation, notebooks, and a controlled API endpoint. The paper argues that problem-first semantic artifact generation can improve reproducibility, inspection, and communication between high-level problem owners and quantum software practitioners while preserving framework-level control for Qiskit and OpenQASM users.

## KEYWORDS

quantum software engineering; OpenQASM; Qiskit; reproducibility; semantic workflows; SDK; problem-first computation

## I. INTRODUCTION

Quantum software engineering increasingly requires interfaces that can connect problem owners, software developers, circuit experts, simulators, and hardware-oriented workflows. In many current workflows, the first practical artifact is a circuit or circuit template. This is useful for execution, but it can also compress or obscure the higher-level structure of the original problem: what the candidates are, what signals or predicates are relevant, which constraints define the allowed state space, what kind of output is expected, and which assumptions were made before circuit materialization.

QDSV Bridge addresses this software-engineering gap by introducing a problem-first semantic-to-artifact layer. Its goal is not to replace Qiskit, OpenQASM, Amazon Braket, or other execution frameworks. Instead, Bridge sits upstream of them and produces auditable handoff artifacts that can be inspected, adapted, simulated, or executed by framework-specific tools. The public SDK exposes a compact Python and CLI interface for generating QDSV IR summaries, oracle specifications, OpenQASM artifacts, Qiskit-oriented blueprints, Bridge Reports, and expert-oriented construction inputs from controlled semantic problem specifications.

The central design principle is simple: do not force a problem into a prefabricated circuit; derive the circuit artifact or construction inputs from a bounded semantic problem specification. This paper presents QDSV Bridge as a practical workflow artifact for quantum software engineering, with emphasis on reproducibility, traceability, user-facing delivery modes, and OpenQASM/Qiskit interoperability.

## II. PROBLEM-FIRST BRIDGE WORKFLOW

Bridge starts from a controlled problem-family specification rather than from an arbitrary circuit. A specification declares the family, state-space

kind, candidate count or domain structure, semantic signals, goal, target format, backend family, and public limits such as qubit or depth bounds. The family acts as a contract: it defines what kind of problem is allowed, what evidence must be produced, what export targets are supported, and which patterns are excluded.

The public developer preview currently includes bounded semantic marking, semantic signal classification, predicate marking, state similarity, combinatorial relation, and distribution sampling families. These are intentionally bounded. Bridge is not a universal free-form circuit generator and does not accept raw datasets, arbitrary Python execution, production secrets, or unmanaged hardware execution requests. Instead, it accepts compact semantic structures and returns artifacts suitable for review.

The workflow can be summarized as follows:

problem intent -> controlled semantic specification -> Bridge validation/build -> QDSV IR and oracle specification -> OpenQASM/Qiskit-oriented artifact or expert-oriented construction inputs -> Bridge Report.

This handoff pattern preserves separation of concerns. Bridge records the declared problem structure and generates auditable artifacts. Qiskit or another downstream framework remains responsible for framework-specific import, inspection, transpilation, simulation, and execution.

## III. DELIVERY MODES

The SDK exposes four delivery modes, all built on the same semantic contract.

Bridge Use is intended for users who need a simpler starting point. It can generate a problem-derived circuit artifact, a simple explanation, warnings, and a ready-to-run example without asking the user to design a circuit first.

Bridge Build is intended for developers who understand Qiskit, OpenQASM, or quantum workflows. It returns a problem-derived artifact plus OpenQASM/Qiskit-oriented content, oracle specification, IR summary, traceability report, estimated resources, and digests.

Bridge Expert Prepare is intended for expert circuit constructors. It returns semantic construction inputs before final circuit materialization: validated family, ProblemSpec/IR, predicates, target state or goal, variables, constraints, encoding suggestions, measurement suggestions, and information-loss risks.

Bridge Expert Evaluate is intended for expert evaluators. It compares possible materialization variants and returns comparison evidence and traceability metadata.

This mode separation is important for usability. A basic user can obtain a ready-to-inspect artifact, while an expert can stop earlier and use the semantic ingredients to design or compare custom materializations.

## IV. IMPLEMENTATION AND ARTIFACT CONTRACT

QDSV Bridge is distributed as the qdsv-bridge Python package under the MIT license. The base SDK is lightweight and depends on HTTP client functionality. Qiskit remains optional through a version-capped extra, currently declared as `qiskit>=2,<3`, so Bridge can follow Qiskit Ecosystem compatibility guidance without making Qiskit a required runtime dependency for every user.

The public SDK talks to a controlled API endpoint and can also be configured for private or local deployments. Public informational endpoints such as family discovery are open. Value-producing

operations are constrained by payload limits, rate limits, and preview quotas. Bridge accepts compact semantic problem specifications rather than full datasets, which helps keep the public preview bounded and reproducible.

For circuit-oriented targets, Bridge can export QASM2, QASM3, Qiskit-oriented blueprints, oracle specifications, IR summaries, and reports. A typical OpenQASM artifact includes the OpenQASM header, circuit registers, generated operations, an explicit semantic oracle materialization point when appropriate, measurement operations, and comments or digests identifying the Bridge family and artifact lineage. In developer preview, some semantic oracle insertion points are intentionally explicit so downstream users can inspect, complete, replace, or adapt the target-specific realization safely.

The Bridge Report is the public trust document for the workflow. It records the problem and deliverable contract, family and mode, validation results, semantic traceability evidence, generated artifact summary, circuit policy, warnings, resource limits, and cryptographic digests. This makes the artifact more than a standalone circuit text file: it becomes part of a traceable export package.

## V. QISKIT AND OPENQASM INTEROPERABILITY

For Qiskit users, Bridge provides an upstream path from problem specification to inspectable artifacts. The generated OpenQASM 3 content can be loaded into Qiskit when the current environment supports the emitted OpenQASM features. Users can then inspect the circuit, simulate locally with Qiskit Aer, modify the artifact, or route it toward IBM Quantum workflows under their own control.

The practical benefit is not loss of framework control. The benefit is traceability before circuit construction. Qiskit users can review what problem was declared, which state-space role was used, which artifact was generated, what warnings were reported, what semantic traceability evidence was attached, and which digests identify the artifact.

QDSV Bridge is listed in the Qiskit Ecosystem as an SDK project and includes a Qiskit-oriented notebook, documentation site, PyPI package, and public examples. This external listing is not treated as a validation of the full QDSV model, but it is an important software-engineering signal that Bridge fits a recognizable interoperability role around Qiskit and OpenQASM workflows.

Bridge also supports OpenQASM-first handoff to other ecosystems. For example, an Amazon Braket-oriented demo adapts the OpenQASM source into the header style expected by the Braket SDK and runs it on LocalSimulator. This demonstrates that Bridge's public artifact boundary can support provider-neutral inspection before framework-specific execution.

## VI. EVALUATION AND CURRENT EVIDENCE

The current evidence for Bridge is primarily software-engineering evidence rather than a claim of quantum advantage. The public repository includes notebooks demonstrating semantic candidate marking, predicate oracle marking, semantic signal classification, IBM/Qiskit artifact inspection, and Amazon Braket OpenQASM handoff. The SDK includes automated tests for client behavior and documentation builds using the Qiskit Ecosystem Sphinx theme. The public package is available on PyPI, and Qiskit-oriented installation uses an explicit version cap to avoid declaring compatibility with unreleased Qiskit major versions.

Current evidence is summarized as follows: semantic specification is implemented through bounded problem-family contracts; OpenQASM export is demonstrated through QASM2/QASM3 artifacts; Qiskit workflow is demonstrated through a public notebook and Qiskit-oriented install path; Braket handoff is demonstrated through a LocalSimulator OpenQASM example; reproducibility reporting is implemented through Bridge Reports, digests, warnings, and export metadata.

The evaluation focus is therefore reproducibility and workflow clarity. A user can begin from a bounded specification, generate an artifact, inspect the OpenQASM or Qiskit-oriented output, produce a Bridge Report, and preserve the semantic and digest metadata needed to review the handoff later. This is useful for education, technical demos, early-stage workflow design, and communication between problem owners and quantum developers.

## VII. LIMITATIONS

Bridge should not be interpreted as a hardware execution SDK, production quantum compiler, unrestricted circuit generator, or proof of quantum advantage. It does not expose private QDSV Runtime internals, backend-routing heuristics, CAP internals, private lowering internals, production adapters, or hardware credentials. It does not

accept raw datasets, unbounded problem descriptions, or arbitrary user-supplied circuits as family contracts.

The current developer preview emphasizes bounded semantic families and inspectable artifact generation. More advanced native lowering, backend routing, managed hardware execution, and provider-specific production integrations remain outside the public SDK. These boundaries are intentional: they keep the public Bridge contract understandable, auditable, and safe for early community use.

## VIII. CONCLUSION

QDSV Bridge contributes a problem-first semantic-to-OpenQASM workflow layer for quantum software engineering. By separating problem specification, artifact generation, framework handoff, and reproducibility reporting, Bridge helps preserve problem intent before circuit-level implementation while keeping downstream control with tools such as Qiskit. The current public preview demonstrates how a lightweight SDK, controlled family contracts, delivery modes, OpenQASM artifacts, and Bridge Reports can support reproducible and auditable quantum-oriented workflows. Future work includes expanding supported families, improving framework-specific examples, strengthening community documentation, and studying how semantic artifact contracts can support larger quantum software engineering pipelines.

## REFERENCES

- [1] J. A. Jimenez Lozano, "QDSV: Quantum Declarative Semantic Value," arXiv:2606.15027, 2026.
- [2] J. A. Jimenez Lozano, "QDSV Framework Implementation and Validation," arXiv:2606.19312, 2026.
- [3] QDSV Bridge repository, <https://github.com/qdsvquantum-afk/qdsv-bridge>
- [4] QDSV Bridge documentation, <https://qdsvquantum-afk.github.io/qdsv-bridge/>
- [5] OpenQASM specification, <https://openqasm.com/>
- [6] Qiskit Ecosystem project page for QDSV Bridge, <https://qiskit.github.io/ecosystem/p/e8734f93/>
- [7] IBM Quantum and Qiskit documentation, <https://docs.quantum.ibm.com/>
- [8] Amazon Braket developer guide, <https://docs.aws.amazon.com/braket/>

## AI-ASSISTED PREPARATION DISCLOSURE

AI-assisted tools were used for drafting support and document preparation. The author reviewed, edited, and takes responsibility for the technical content, claims, and final submission.