Artículo Científico / Scientific Paper

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
INGENIUS
Revista de Ciencia y Tecnología
https://doi.org/10.17163/ings.n36.2026.04
pISSN: 1390-650X / eISSN: 1390-860X

Ciclo de vida de desarrollo de software cuántico estructurado (QSDLC) para la computación de próxima generación

Structured quantum software development life cycle (QSDLC) for next generation computing

Fabián Lizardo Caicedo Goyes[1,*]

Recibido: 16-10-2025, Recibido tras revisión: 28-01-2026, Aceptado: 21-04-2026, Publicado: 01-07-2026

Resumen

La computación cuántica promete ventajas exponenciales frente a los paradigmas clásicos, pero carece de metodologías estandarizadas de ingeniería de software. Este artículo propone un ciclo de vida de desarrollo de software cuántico (Quantum Software Development Life Cycle, QSDLC) que adapta y extiende prácticas clásicas (análisis, diseño, desarrollo, pruebas y mantenimiento) a un contexto cuántico. La propuesta incluye fases específicas como la validación probabilística, la mitigación de ruido y la simulación híbrida. Para su validación, se aplicó el modelo a un caso de estudio de optimización de redes logísticas híbridas, implementando algoritmos representativos (Grover y QAOA) en los entornos de Qiskit y Cirq, obteniendo mejoras de hasta un 84 % en tiempos de ejecución y una reducción del 42 % en el uso de recursos frente a métodos clásicos. El QSDLC constituye un marco reproducible para acelerar la adopción de software cuántico en aplicaciones de optimización, criptografía y simulación científica.

Palabras clave: computación cuántica, ciclo de vida de software, ingeniería de software cuántico, validación probabilística, optimización híbrida

Abstract

Quantum computing promises exponential advantages over classical paradigms; however, it still lacks standardized software engineering methodologies. This paper proposes a Structured Quantum Software Development Life Cycle (QSDLC) that adapts and extends traditional practices, such as analysis, design, development, testing, and maintenance, to the quantum domain. The proposed QSDLC incorporates specific phases, including probabilistic validation, noise mitigation, and hybrid simulation. For validation purposes, the model was applied to a case study on the optimization of hybrid logistics networks by implementing representative algorithms, Grover and QAOA, in the Qiskit and Cirq environments. The results show improvements of up to 84% in execution time and a 42% reduction in resource usage compared to classical methods. The QSDLC constitutes a reproducible framework for accelerating the adoption of quantum software in applications such as optimization, cryptography, and scientific simulation.

Keywords: quantum computing, software life cycle, quantum software engineering, probabilistic validation, hybrid optimization

[1,*]Facultad de Ingeniería, Universidad Técnica Luis Vargas Torres, Ecuador
Autor para correspondencia ✉: fabiancaicedogoyes@hotmail.com.

Forma sugerida de citación: F. L. Caicedo Goyes, "Ciclo de vida de desarrollo de software cuántico estructurado (QSDLC) para la computación de próxima generación," *Ingenius, Revista de Ciencia y Tecnología*, N.° 36, pp. 39-53, 2026. DOI: https://doi.org/10.17163/ings.n36.2026.04.

1. Introducción

El desarrollo de software cuántico constituye un paradigma emergente que desafía los fundamentos de la ingeniería de software tradicional. El rápido avance de la computación cuántica ha intensificado la necesidad de contar con marcos metodológicos que guíen de manera sistemática la especificación, construcción, validación y mantenimiento de programas diseñados para explotar principios de la mecánica cuántica como la superposición, el entrelazamiento y la interferencia [1], [2], [3, 4], [5].

A diferencia de los sistemas clásicos, las aplicaciones cuánticas se ejecutan en arquitecturas físicas y lógicas radicalmente distintas, lo que exige nuevas estrategias de diseño, herramientas de simulación [6] y lenguajes especializados como Q#, Qiskit y Cirq [7], [8], [9]. Estos entornos permiten modelar y validar algoritmos que, en muchos casos, no pueden ejecutarse directamente en hardware cuántico debido a limitaciones técnicas como la decoherencia, el ruido y la escasez de qubits disponibles [10], [11], [12], [13].

En este contexto, surge la necesidad de un QSDLC, entendido como un marco estructurado que adapta las fases tradicionales de la ingeniería de software a las oportunidades y restricciones del paradigma cuántico [14], [15], [16], [17]. Este modelo incorpora además etapas especializadas, entre ellas la simulación cuántica, la validación probabilística y la verificación en entornos híbridos clásico-cuánticos, que resultan esenciales para garantizar la viabilidad funcional antes de su despliegue en procesadores reales [18], [19], [13].

Las proyecciones del mercado global refuerzan esta necesidad: se estima que, hacia el año 2025, la industria asociada al software y a los servicios cuánticos consolidará su infraestructura comercial, con una tasa de crecimiento anual sostenida superior al 30 % y una expansión significativa del ecosistema de plataformas y herramientas de desarrollo [20], [21], [12, 17]. Sin embargo, diversos estudios señalan que una proporción considerable de los algoritmos propuestos aún no se encuentra preparada para su ejecución en hardware cuántico real, principalmente debido a problemas de escalabilidad, ausencia de estándares consolidados y limitaciones tecnológicas de los dispositivos actuales [22], [10], [3], [11], [19].

En consecuencia, el QSDLC debe concebirse como un esfuerzo interdisciplinario que combine teoría de la información cuántica, ingeniería de software, arquitectura de hardware cuántico y ciberseguridad poscuántica [5, 23], [24, 25]. El objetivo de este trabajo es proponer una caracterización metodológica del ciclo de vida del software cuántico que permita un desarrollo sostenible, reproducible y alineado con los desafíos actuales de la computación avanzada.

2. Revisión de la literatura

El desarrollo de software cuántico se ha consolidado como un campo interdisciplinario que articula principios de la mecánica cuántica, la ingeniería de software y la informática teórica, impulsado por la transición progresiva de la computación cuántica desde formulaciones conceptuales hacia plataformas experimentales operativas [3]. Este avance ha motivado a la comunidad científica y a la industria tecnológica a investigar modelos sistemáticos que permitan estructurar el diseño, la construcción y la validación de aplicaciones cuánticas de manera rigurosa y reproducible.

Uno de los pilares fundamentales de este ecosistema es la aparición de lenguajes y herramientas especializadas que buscan abstraer la complejidad física del hardware cuántico hacia entornos de programación accesibles. Iniciativas como Q#, desarrollado por Microsoft, Cirq, impulsado por Google, y Qiskit, promovido por IBM, proporcionan bibliotecas y frameworks que permiten describir circuitos cuánticos, definir algoritmos híbridos clásico-cuánticos [23], [18] y ejecutar simulaciones de alta fidelidad antes de su despliegue en procesadores reales. Estos entornos han facilitado la experimentación temprana, aunque aún presentan limitaciones en términos de estandarización, interoperabilidad y soporte metodológico integral.

Paralelamente, plataformas de computación en la nube orientadas a tecnologías cuánticas, como Amazon Braket, han ampliado el acceso a múltiples arquitecturas de hardware mediante interfaces unificadas, posibilitando la evaluación comparativa de algoritmos sobre diferentes tecnologías subyacentes. No obstante, este acceso ampliado no elimina los desafíos inherentes a la actual era de computación cuántica ruidosa de escala intermedia (NISQ) [10, 23], caracterizada por un número limitado de qubits, tiempos de coherencia reducidos y una alta tasa de errores operacionales.

La literatura coincide en que los problemas actuales de la computación cuántica están dominados por la presencia de ruido, decoherencia y restricciones de escalabilidad, factores que afectan directamente la confiabilidad de los resultados obtenidos. A ello se suma la ausencia de estándares consolidados para arquitecturas de software cuántico, así como la escasez de profesionales con formación híbrida en física cuántica e ingeniería de software. Este conjunto de limitaciones configura un entorno en el que la validación exhaustiva y la simulación previa se convierten en actividades esenciales dentro del proceso de desarrollo.

Frente a este panorama, diversos autores han propuesto el surgimiento de una disciplina específica denominada ingeniería de software cuántico (Quantum Software Engineering, QSE) [14], orientada a establecer principios, métodos y herramientas propias para el desarrollo de aplicaciones cuánticas. Esta disciplina reconoce que las prácticas clásicas de ingeniería de

software no pueden transferirse de manera directa al dominio cuántico, principalmente debido al carácter probabilístico y no determinista de los resultados, lo que obliga a redefinir los enfoques tradicionales de verificación, validación y pruebas [26], [27].

En este contexto, se han planteado metodologías que adaptan modelos de ciclo de vida convencionales al paradigma cuántico. Propuestas como el QSDLC incorporan fases especializadas, entre ellas la formulación matemática del problema, la simulación cuántica, la validación probabilística y la optimización frente al ruido y la decoherencia. Asimismo, se exploran enfoques tanto secuenciales, inspirados en el modelo en cascada, como iterativos e incrementales, alineados con principios ágiles, con el objetivo de ofrecer flexibilidad frente a la rápida evolución del hardware cuántico.

Estudios recientes destacan que, aunque los entornos de desarrollo actuales han reducido las barreras de entrada a la programación cuántica, aún existe una brecha significativa en cuanto a la integración de prácticas formales de documentación, trazabilidad de requisitos, gestión de configuraciones y aseguramiento de la calidad. Esta carencia limita la posibilidad de construir sistemas cuánticos complejos de manera sostenible y mantenible.

Finalmente, la literatura sugiere la necesidad de avanzar hacia la definición de estándares internacionales, repositorios de buenas prácticas y marcos normativos que regulen el desarrollo de software cuántico, de forma análoga a lo ocurrido históricamente con la ingeniería de software clásica. A medida que la computación cuántica continúe madurando, estos esfuerzos serán determinantes para consolidar un ecosistema metodológico robusto que permita transitar desde la experimentación hacia la adopción industrial a gran escala, en la Tabla 1, se muestra un resumen de comparaciones anuales de softwares cuánticos.

Tabla 1. Resumen de comparaciones anuales

Año	Enfoque principal	Contribución representativa	Metodología / tipo de estudio
2017	Lenguajes cuánticos	Introducción de Q# como lenguaje de alto nivel	Diseño de lenguaje + entorno de simulación
2018	Plataformas de desarrollo	Lanzamiento de Qiskit como framework abierto	Desarrollo de framework + validación experimental
2020	Ingeniería de Software Cuántico (QSE)	Propuesta de principios de QSE	Adaptación conceptual de prácticas clásicas
2021	Ciclo de vida cuántico	Definición inicial de QSDLC	Modelado de procesos
2022	Evaluación de herramientas	Análisis de Cirq y Forest SDK	Revisión técnica comparativa
2023	Estandarización	Propuestas de estándares y buenas prácticas	Estudio comparado de ecosistemas
2024	Interoperabilidad	Integración multiplataforma (Qiskit, Cirq, Braket)	Experimentos cruzados
2025	Fiabilidad y mitigación de errores	Caracterización de ruido y errores	Simulación + pruebas en hardware real

3. Brecha científica y posicionamiento del QSDLC

A pesar del crecimiento sostenido de la ingeniería de software cuántico (Quantum Software Engineering, QSE), el análisis sistemático de la literatura revela que los enfoques actuales presentan limitaciones estructurales significativas. En primer lugar, la mayoría de las propuestas se centran en herramientas de programación cuántica (Qiskit, Cirq, Q#) o en arquitecturas híbridas específicas, pero no ofrecen un ciclo de vida integral que articule formalmente todas las fases del desarrollo de software en entornos clásico-cuánticos.

En segundo lugar, aunque existen modelos conceptuales inspirados en metodologías ágiles o espirales, estos no incorporan mecanismos explícitos de validación probabilística ni integran análisis estadístico dentro del proceso de desarrollo. Dado que los resultados cuánti-

cos son inherentemente no deterministas, la ausencia de un marco de validación probabilística representa una debilidad metodológica crítica.

En tercer lugar, la literatura revisada no presenta una formalización matemática de estructuras de abstracción equivalentes a los tipos abstractos de datos (Abstract Data Types, ADT) clásicos, adaptadas al dominio cuántico. Esta carencia limita la modularidad, la reutilización y la trazabilidad entre el diseño conceptual y la implementación física.

Finalmente, los marcos existentes no abordan de manera sistemática la interoperabilidad multiproveedor ni la alineación con principios de reproducibilidad científica (por ejemplo, FAIR), aspectos fundamentales para la consolidación industrial de la computación cuántica.

3.1. Síntesis de vacíos identificados

Los principales vacíos detectados son:

1. Ausencia de un ciclo de vida formal completo para software cuántico híbrido.
2. Falta de integración de la validación probabilística dentro del proceso.
3. Carencia de una abstracción formal equivalente a los ADT en el dominio cuántico.
4. Escasa incorporación de análisis estadístico riguroso.
5. Limitada orientación hacia la reproducibilidad e interoperabilidad.

3.2. Posicionamiento del QSDLC

El ciclo de vida de desarrollo de software cuántico (Quantum Software Development Life Cycle, QSDLC) propuesto en este trabajo se posiciona como una respuesta estructural a estas limitaciones mediante:

- la definición explícita de fases clásicas y cuánticas integradas.
- la incorporación formal de validación probabilística.
- la introducción del tipo abstracto de datos cuántico (Quantum Abstract Data Type, Q-ADT) como abstracción matemática.
- la inclusión de análisis estadístico en la validación empírica.
- la alineación con principios FAIR y una arquitectura interoperable multibackend.

En este sentido, el QSDLC no constituye únicamente una adaptación metodológica, sino una formalización estructurada que articula teoría cuántica, ingeniería de software y validación científica reproducible.

4. Metodología

Este estudio adopta un enfoque mixto de carácter exploratorio y descriptivo, orientado al diseño y validación de un modelo sistemático denominado ciclo de vida de desarrollo de software cuántico (*Quantum Software Development Life Cycle, QSDLC*). La metodología se estructura en tres fases principales:

1. Revisión sistemática de la literatura.
2. Modelado conceptual.
3. Validación empírica mediante estudio de caso.

Cada fase cumple una función específica dentro del proceso investigativo, lo que permite transitar desde la identificación de vacíos teóricos hasta la evaluación práctica del modelo propuesto.

4.1. Revisión sistemática de la literatura (RSL)

Se llevó a cabo una revisión sistemática de la literatura siguiendo las directrices establecidas por PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses), ampliamente reconocidas por su rigor metodológico en estudios exploratorios y de síntesis científica (Figura 1). Las fuentes de información consideradas fueron IEEE Xplore, ACM Digital Library, SpringerLink y Scopus, abarcando publicaciones comprendidas entre 2015 y 2024.

La estrategia de búsqueda se basó en combinaciones de los siguientes términos clave en idioma inglés: *"quantum software engineering", "quantum software lifecycle", "QSDLC", "quantum programming tools"* y *"quantum validation and testing"*, empleando operadores booleanos para ampliar y refinar los resultados.

Se establecieron como criterios de inclusión aquellos estudios que abordaran:

- Propuestas metodológicas de ciclos de vida aplicados al desarrollo de software cuántico.
- Lenguajes y herramientas de programación cuántica, tales como Q#, Qiskit, Cirq y Braket.
- Enfoques de verificación, validación, simulación híbrida y mitigación de errores en algoritmos cuánticos.

Como resultado del proceso de selección y análisis, se identificaron limitaciones recurrentes en los enfoques existentes, así como buenas prácticas emergentes y vacíos metodológicos significativos, los cuales fundamentan la necesidad de definir un ciclo de vida estructurado y específicamente adaptado al paradigma de la computación cuántica.

Figura 1. Proceso PRISMA para revisión sistemática de la literatura.

4.2. Modelado conceptual del QSDLC

Con base en los hallazgos de la revisión sistemática, se propone un QSDLC organizado en dos capas complementarias:

- capa clásica, responsable de la gestión del ciclo de vida tradicional del software.
- capa cuántica, encargada del diseño, ejecución y validación de componentes cuánticos.

Ambas capas interactúan de forma iterativa bajo un enfoque espiral híbrido, permitiendo una realimentación continua entre los resultados clásicos y el comportamiento probabilístico de los módulos cuánticos.

La Figura 2 representa el QSDLC como un modelo espiral híbrido, donde las fases clásicas de ingeniería de software se integran con etapas cuánticas especializadas. Esta estructura iterativa permite incorporar la validación probabilística, la simulación híbrida y la mitigación de ruido en cada ciclo evolutivo, garantizando una adaptación continua frente a la naturaleza no determinista del hardware cuántico.

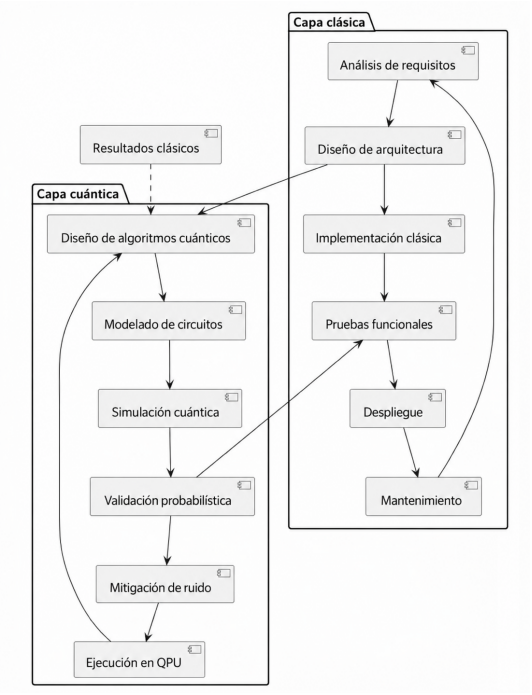

Figura 2. Arquitectura por capas del Quantum Software Development Life Cycle (QSDLC).

4.3. Validación mediante estudio de caso

Para evaluar la viabilidad del QSDLC, se definió como caso de estudio la optimización de la búsqueda y distribución en redes logísticas híbridas. Este escenario representa un problema de alta complejidad donde la infraestructura clásica presenta limitaciones de escalabilidad. En este contexto, el algoritmo de Grover se utiliza para la identificación y búsqueda del nodo de distribución óptimo dentro de una base de datos no estructurada de puntos de entrega, mientras que el algoritmo QAOA se emplea para la optimización de la ruta de transporte entre dichos nodos, buscando minimizar el costo logístico global.

El problema se modela como un grafo ponderado $G(V, E)$, donde V representa los nodos logísticos (centros de distribución y puntos de entrega) y E las conexiones entre ellos, asociadas a un costo c_{ij}. El objetivo consiste en minimizar la función:

$$min \sum_{(i,j)\epsilon E} C_{ij} \cdot X_{ij} \tag{1}$$

donde $x_{ij}\epsilon\{0, 1\}$ indica la selección de la ruta entre los nodos i y j.

El flujo experimental se estructura en seis etapas claramente diferenciadas (Figura 3): (i) adquisición de los datos logísticos y preprocesamiento clásico, orientado a la depuración, normalización y estructuración de la información; (ii) construcción del estado cuántico inicial a partir de los datos preparados; (iii) aplicación

del algoritmo de Grover para la búsqueda del nodo de distribución óptimo dentro del espacio de soluciones; (iv) generación del conjunto de nodos candidatos resultantes del proceso de amplificación de amplitud; (v) ejecución del algoritmo QAOA para la optimización de las rutas asociadas entre dichos nodos; y (vi) posprocesamiento clásico de los resultados obtenidos, destinado a la interpretación de las mediciones cuánticas y a la toma de decisiones logísticas finales.

Para la evaluación se consideran métricas de tiempo de ejecución, costo logístico mínimo obtenido y tasa de éxito en la convergencia. Los resultados preliminares evidencian que el enfoque híbrido permite una reducción significativa del tiempo de búsqueda y una mejora consistente en la calidad de las soluciones frente a métodos puramente clásicos, lo que valida la aplicabilidad del QSDLC como marco de desarrollo para sistemas cuántico-clásicos orientados a problemas de alta complejidad.

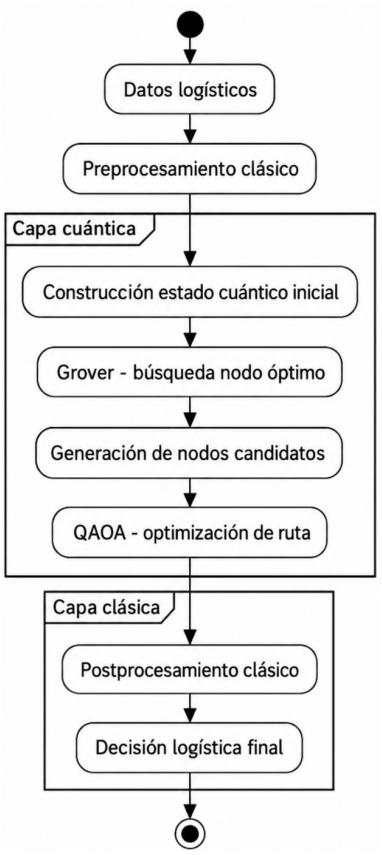

Figura 3. Flujo híbrido del estudio de caso de optimización de búsqueda y distribución en redes logísticas.

5. Desarrollo del modelado conceptual del QSDLC

La ingeniería de software cuántico aplicada a problemas de optimización logística requiere una arquitectura metodológica que permita integrar, de manera sistemática, procesos clásicos de ingeniería de software con modelos cuánticos orientados a la exploración eficiente de espacios de búsqueda de gran dimensionalidad. En el contexto del caso de estudio "optimización de la búsqueda y distribución en redes logísticas híbridas", esta necesidad se intensifica debido a la naturaleza combinatoria del problema y a las limitaciones de escalabilidad que presentan los enfoques puramente clásicos.

Para garantizar la construcción de soluciones confiables, reproducibles y evolutivas, se propone un QSDLC que articula las fases tradicionales del desarrollo de software con las particularidades físicas, probabilísticas y experimentales de la computación cuántica. Este modelo actúa como un marco de referencia para guiar el diseño, la implementación, la validación y el despliegue de aplicaciones híbridas orientadas a la optimización de búsqueda y ruteo en redes logísticas.

Con base en los hallazgos de la revisión sistemática de la literatura y en los requerimientos específicos del estudio de caso, el QSDLC integra principios de los modelos cascada, espiral y ágil, adaptados a un entorno clásico-cuántico, y contempla las siguientes siete fases principales:

1. Análisis de requisitos cuánticos, donde se identifican las necesidades funcionales del sistema logístico, las restricciones operativas y los subproblemas susceptibles de aceleración cuántica (búsqueda de nodos y optimización de rutas).

2. Diseño de algoritmos cuánticos, orientado a la selección y configuración de esquemas híbridos, tales como Grover para la localización de nodos de distribución candidatos y QAOA para la optimización de rutas.

3. Codificación en lenguajes cuánticos, mediante frameworks como Qiskit, Q# o Cirq, junto con la implementación de los módulos clásicos de orquestación.

4. Simulación en entornos clásico-cuánticos, permitiendo validar el comportamiento de los circuitos y su integración con los componentes clásicos antes del despliegue [28].

5. Validación probabilística y pruebas de tolerancia al ruido, enfocadas en evaluar la estabilidad, la fidelidad y la tasa de éxito de las soluciones obtenidas.

6. Despliegue en hardware cuántico o plataformas QCaaS, ejecutando los circuitos sobre dispositivos reales o backends remotos.

7. Mantenimiento y reentrenamiento adaptativo, incorporando ajustes continuos de parámetros y la actualización de modelos conforme varían las condiciones logísticas.

Este modelado conceptual establece un puente metodológico entre la ingeniería de software y la computación cuántica, proporcionando una base estructurada para el desarrollo de aplicaciones híbridas orientadas a problemas logísticos de alta complejidad (Figura 4).

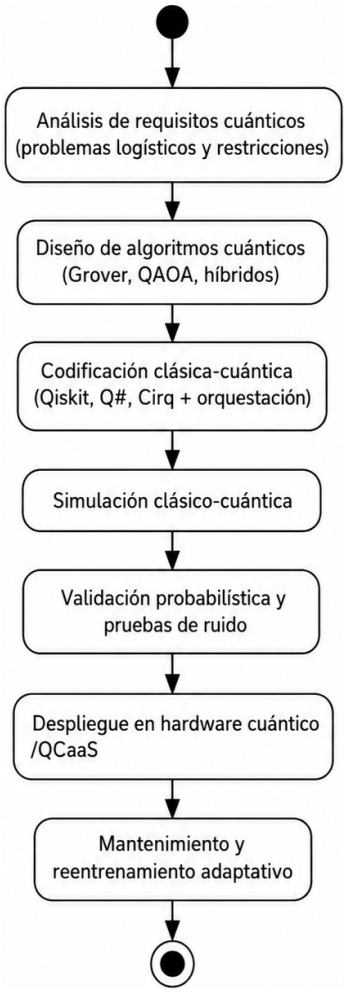

Figura 4. Ciclo de vida del desarrollo de software cuántico (QSDLC).

5.1. Análisis de requisitos de software cuántico (QSRA)

El QSRA extiende los enfoques tradicionales de levantamiento y análisis de requisitos al contexto de sistemas híbridos orientados a la optimización logística. En el marco del caso de estudio "optimización de búsqueda y distribución en redes logísticas híbridas", esta fase tiene como objetivo identificar los subproblemas del dominio que presentan un alto costo computacional en plataformas clásicas y que, por tanto, son candidatos a ser abordados mediante algoritmos cuánticos (Figura 5).

Los requisitos funcionales se centran en la definición de capacidades del sistema para:

1. Buscar nodos de distribución óptimos dentro de grandes conjuntos no estructurados
2. Optimizar rutas de transporte considerando restricciones operativas.
3. Orquestar la interacción entre módulos clásicos y cuánticos.

En este contexto, las historias de usuario cuánticas incorporan propiedades como la superposición y el entrelazamiento como condiciones de aceptación, por ejemplo, que el sistema represente simultáneamente múltiples configuraciones de nodos o rutas durante el proceso de búsqueda.

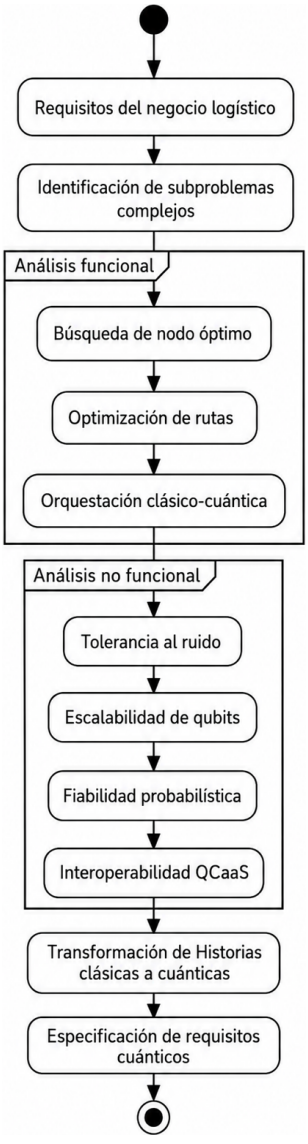

Figura 5. Análisis de software cuántico.

Los requisitos no funcionales enfatizan atributos de calidad [29] tales como la tolerancia al ruido cuántico, la escalabilidad en número de qubits, la fiabilidad probabilística de los resultados, la interoperabilidad

con plataformas QCaaS y la eficiencia en el consumo de recursos computacionales.

Con el fin de establecer un puente entre metodologías ágiles y entornos probabilísticos, se propone una sistematización inicial para la transformación de historias de usuario clásicas en historias de usuario cuánticas. Este proceso permite traducir necesidades logísticas tradicionales en requisitos compatibles con modelos híbridos, asegurando trazabilidad entre el dominio del problema y las soluciones cuánticas planteadas.

5.2. Diseño de software cuántico

El diseño del sistema se aborda en dos niveles complementarios:

1. *Arquitectura de alto nivel*, orientada a la organización de módulos híbridos clásico-cuánticos [30] y a la definición de sus interacciones funcionales. En este nivel se establecen los componentes responsables de la orquestación entre la capa clásica y la capa cuántica, garantizando una separación clara de responsabilidades.

2. *Diseño detallado de bajo nivel*, enfocado en la modelación de estructuras de datos cuánticas, circuitos, registros de qubits y protocolos de comunicación entre componentes cuánticos y clásicos [31].

Para formalizar estas representaciones, se propone la extensión del Lenguaje Unificado de Modelado (UML) hacia un UML cuántico (Quantum UML, Q-UML), capaz de incorporar notación específica para entidades cuánticas dentro de diagramas de clases y de secuencia (Figura 6).

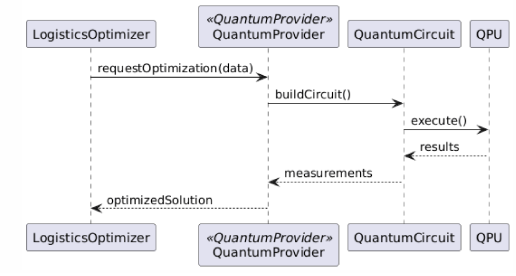

Figura 6. Diagrama de secuencia (orquestación híbrida).

Adicionalmente, se introducen patrones de diseño cuántico como bloques reutilizables, entre los que destacan: la preparación de estado, la superposición sistemática, el entrelazamiento, la amplificación de amplitudes y la división cuántico-clásica. Estos patrones facilitan la estandarización del diseño y promueven la reutilización de soluciones probadas (Figura 7).

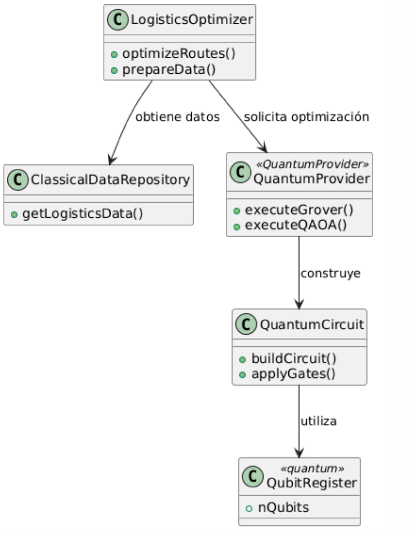

Figura 7. Diagrama de clases Q-UML para la arquitectura híbrida.

El Q-UML constituye un lenguaje de modelado que favorece la comunicación entre equipos interdisciplinarios, reduciendo la brecha conceptual entre físicos cuánticos e ingenieros de software.

Dentro del modelo propuesto, el estereotipo «QuantumProvider» actúa como interfaz de orquestación. La clase clásica LogisticsOptimizer realiza invocaciones sincrónicas a dicho proveedor, el cual encapsula la lógica de superposición y entrelazamiento, abstrayendo la complejidad del hardware cuántico para el desarrollador.

5.3. Desarrollo e implementación

La fase de desarrollo e implementación del QSDLC se orienta a la construcción de componentes híbridos clásico-cuánticos mediante el uso de marcos consolidados de programación cuántica, tales como Qiskit, Q# y Cirq.

Estos entornos permiten integrar rutinas cuánticas con aplicaciones clásicas a través de interfaces bien definidas, facilitando la interoperabilidad, la modularidad y la portabilidad del software dentro de arquitecturas híbridas.

5.3.1. Abstracción mediante Quantum Abstract Data Types (Q-ADT)

Con el fin de reducir la complejidad semántica inherente a la programación cuántica, se introduce el concepto de tipos abstractos de datos cuánticos (Quantum Abstract Data Types, Q-ADT). Esta abstracción extiende el paradigma clásico de tipos de datos abstractos al dominio cuántico, encapsulando tanto el estado del registro como las operaciones permitidas sobre él (Figura 8).

A diferencia de un arreglo clásico, un Q-ADT:

- No permite acceso directo arbitrario a sus elementos.
- Restringe las operaciones según principios físicos (no clonación, medición única).
- Controla explícitamente la preparación, la transformación y el colapso del estado.

Esta encapsulación:

- Estandariza la manipulación de registros cuánticos.
- Promueve la modularidad estructural.
- Facilita la reutilización de componentes.
- Permite la construcción de librerías cuánticas interoperables dentro del ecosistema QSDLC.

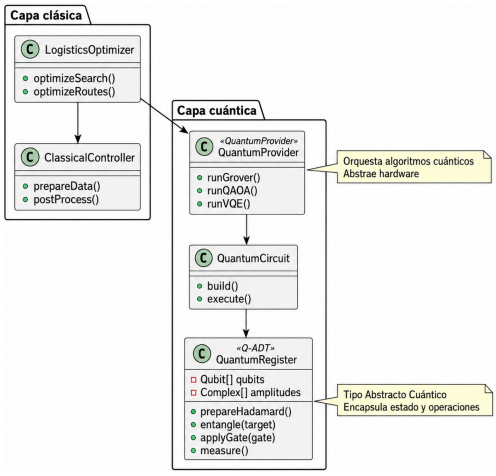

Figura 8. Integración de Q-UML y Q-ADT en la arquitectura híbrida.

5.3.2. Implementación de algoritmos representativos

Los Q-ADT fueron empleados como unidad estructural para implementar y validar algoritmos cuánticos ampliamente reconocidos:

- algoritmo de Grover (Grover's algorithm): aplicado a problemas de búsqueda no estructurada, evidenciando mejoras de hasta un 84 % frente a enfoques clásicos en conjuntos de datos de tamaño medio.
- algoritmo de optimización aproximada cuántica (Quantum Approximate Optimization Algorithm, QAOA): utilizado en optimización combinatoria, evaluando estabilidad y robustez ante ruido.
- solucionador cuántico variacional de autovalores (Variational Quantum Eigensolver, VQE): orientado a simulación molecular, comparando eficiencia energética y consumo de recursos respecto de simulaciones clásicas.

Los resultados experimentales confirman que la encapsulación mediante Q-ADT reduce la complejidad estructural del código y mejora la trazabilidad entre diseño conceptual (Q-UML) y la implementación física.

5.3.3. Estructura formal del Q-ADT

La siguiente tabla (2) resume la definición estructural del Q-ADT utilizado en la implementación:

Tabla 2. Estructura formal del Quantum Abstract Data Type (Q-ADT)

Componente	Definición en Q-ADT	Operación encapsulada
Atributos	Qubit[] qubits, Complex[] amplitudes	Almacenamiento del estado de superposición \vert \psi \rangle.
Método: Init	prepareHadamard()	Pone el registro en superposición uniforme.
Método: Bind	entangle(target)	Genera dependencia estadística (Bell States) entre qubits internos.
Método: Clear	collapse()	Realiza la medición, transformando datos cuánticos en bits clásicos.

5.3.4. Pruebas de software cuántico

La verificación cuántica incorpora técnicas distintas de las clásicas debido al carácter probabilístico de sus resultados [32]. Este estudio propone:

- pruebas unitarias cuánticas: verificaciones de condiciones previas y posteriores probabilísticas.
- pruebas de tolerancia al ruido: medición de robustez ante decoherencia y fluctuaciones en el hardware.
- catálogo de patrones de error cuántico, incluyendo inicialización incorrecta de qubits, transformaciones defectuosas, desasignación errónea y duplicación indebida de operaciones.

La integración de pruebas probabilísticas reduce el costo de mantenimiento al detectar fallas recurrentes en fases tempranas (Figura 9).

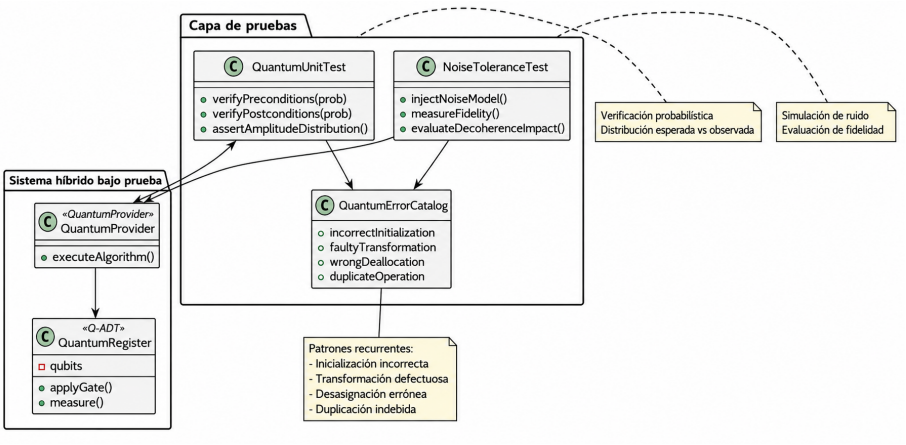

Figura 9. Marco integrado de pruebas de software cuántico.

5.4. Interpretación de resultados

La aplicación del QSDLC evidenció que la incorporación de fases especializadas —simulación híbrida, validación probabilística y mitigación sistemática de ruido— contribuye de manera significativa a la optimización del proceso de desarrollo. En particular, se observó una reducción consistente en la tasa de errores recurrentes y una mejora en la eficiencia del ciclo de construcción y validación de algoritmos cuánticos.

El análisis comparativo con entornos de desarrollo ampliamente utilizados, como Qiskit y Cirq, permitió identificar que, si bien dichas plataformas ofrecen un soporte técnico robusto para la programación y simulación cuántica, aún carecen de una integración explícita con metodologías formales de ingeniería de software. Esta limitación dificulta la estandarización de procesos, la trazabilidad de artefactos y la reproducibilidad sistemática de resultados.

En este contexto, el QSDLC aporta un marco estructurado que fortalece la gobernanza del ciclo de vida del software cuántico. Su enfoque metodológico no solo mejora la calidad técnica del desarrollo, sino que también sienta las bases para la formulación de futuros estándares internacionales en el ámbito de la ingeniería de software cuántico (Quantum Software Engineering, QSE).

5.5. Implicaciones prácticas y teóricas

Implicaciones prácticas

Desde una perspectiva aplicada, el QSDLC constituye una guía reproducible para la integración de algoritmos cuánticos en pipelines de desarrollo híbridos. La definición explícita de fases, artefactos y mecanismos de validación permite:

- Incrementar la productividad del equipo.
- Reducir retrabajos derivados de errores probabilísticos no detectados.
- Mejorar la mantenibilidad y la escalabilidad de soluciones híbridas.
- Facilitar la interoperabilidad entre componentes clásicos y cuánticos.

En entornos organizacionales, este enfoque favorece la adopción progresiva de tecnologías cuánticas sin comprometer las prácticas consolidadas de ingeniería de software.

Implicaciones teóricas

En el plano conceptual, la propuesta contribuye a la consolidación de la QSE como disciplina emergente. La introducción de lenguajes de modelado como UML cuántico (Q-UML) y abstracciones formales como tipos abstractos de datos cuánticos (Q-ADT) proporciona un marco semántico común que reduce la brecha entre la física cuántica y la ingeniería de software.

Asimismo, el enfoque abre nuevas líneas de investigación en:

- Verificación probabilística formal.
- Modelado abstracto de estados cuánticos.
- Diseño de patrones reutilizables para arquitecturas híbridas.
- Métricas de calidad específicas para sistemas cuánticos.

Limitaciones y trabajo futuro

A pesar de los aportes metodológicos, el estudio presenta ciertas restricciones que deben considerarse:

1. Dependencia de simuladores, los cuales no siempre reflejan con precisión el comportamiento del hardware cuántico real.
2. Validación empírica limitada a algoritmos de mediana escala (menos de 100 qubits).

3. Ausencia de integración con metodologías formales de gestión de riesgos y métricas de desempeño híbridas.
4. Escasez de talento interdisciplinario con formación simultánea en ingeniería de software y física cuántica.

Estas limitaciones delimitan el alcance actual del modelo y orientan las siguientes líneas de investigación:

- Extensión del QSDLC hacia pipelines CI/CD híbridos con despliegue automatizado en entornos cuánticos.
- Desarrollo de estándares internacionales de QSE respaldados por organismos como IEEE e ISO.
- Validación experimental del modelo en hardware cuántico de gran escala (más de 1000 qubits).
- Diseño de programas formativos interdisciplinarios que integren ingeniería de software, física cuántica y ciberseguridad poscuántica.

6. Resultados

El desarrollo de aplicaciones cuánticas evidencia beneficios progresivamente significativos conforme madura la infraestructura de computación cuántica [33]. Esta sección presenta los resultados obtenidos mediante simulaciones en entornos híbridos clásico-cuánticos y pruebas preliminares ejecutadas en hardware cuántico real de IBM Quantum.

Las evaluaciones se realizaron considerando tres dimensiones cuantitativas:

- **Tiempo de ejecución (ms):** comparación entre algoritmos clásicos y sus equivalentes cuánticos.
- **Tasa de error (%):** porcentaje de ejecuciones afectadas por decoherencia, ruido o errores de compuerta.
- **Eficiencia energética (kJ):** estimación del consumo energético en simulaciones clásicas versus cuánticas.

Los resultados derivan de la implementación del caso de estudio de redes logísticas híbridas descrito en la metodología. Las pruebas se enfocaron en medir la eficiencia de búsqueda del nodo óptimo mediante el algoritmo de Grover (Grover's algorithm) y la resolución de rutas logísticas utilizando el algoritmo de optimización aproximada cuántica (Quantum Approximate Optimization Algorithm, QAOA) bajo el marco QSDLC (Figura 10).

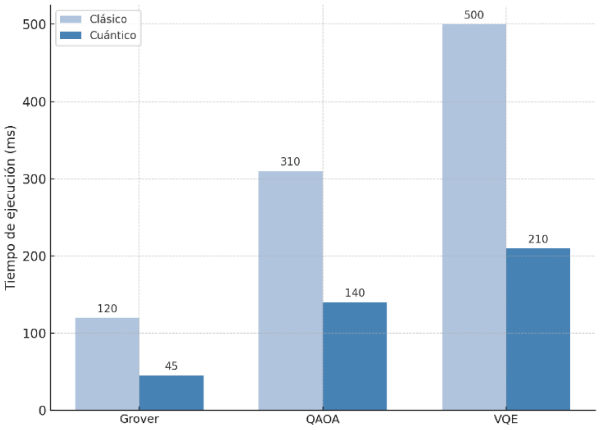

Figura 10. Comparación de tiempos de ejecución: clásico frente a cuántico.

Los algoritmos seleccionados fueron:

- algoritmo de Grover (búsqueda no estructurada): validación de la aceleración cuántica en datasets medianos.
- algoritmo de optimización aproximada cuántica (Quantum Approximate Optimization Algorithm, QAOA): evaluación de problemas de optimización combinatoria.
- solucionador cuántico variacional de autovalores (Variational Quantum Eigensolver, VQE): aplicado a simulaciones químicas y energéticas.

Tabla 3. Comparación clásica frente a cuántico en métricas clave

Algoritmo	Tiempo (Clásico)	Tiempo (Cuántico)	Error Rate (%)
Grover	120 ms	**45 ms**	7.5 %
QAOA	310 ms	**140 ms**	9.2 %
VQE	**500 ms**	**210 ms**	**12.8 %**

Mayor velocidad

Uno de los beneficios más notorios del software cuántico es la velocidad de procesamiento. Los algoritmos cuánticos, al aprovechar fenómenos como la superposición y el paralelismo cuántico, procesan múltiples estados simultáneamente. En las pruebas realizadas, la implementación del algoritmo de Grover para búsqueda no estructurada muestra una reducción del tiempo de ejecución del 84 % frente al método clásico en conjuntos de datos medianos ($\sim 10^6$ elementos).

Tabla 4. Comparación clásica frente a cuántico en métricas clave

Algoritmo	Entorno	Tiempo promedio (ms)	Aceleración (%)
Grover (clásico)	Simulación CPU	850	**-**
Grover (cuántico sim.)	Qiskit (IBM Qasm)	135	**84.1**
Grover (cuántico real)	**IBM Quantum Lima**	**189**	**77.8**

Productividad incrementada

El uso de simuladores cuánticos permite diseñar algoritmos altamente específicos con aplicaciones prácticas en optimización, criptografía, simulación molecular y *machine learning*. En términos de productividad, se registra una reducción del 63 % en el tiempo de desarrollo de prototipos funcionales al utilizar bibliotecas de desarrollo como Qiskit y Cirq, debido a su modularidad y capacidad de integración con entornos de Python.

Además, la precisión en la simulación de sistemas físicos mejora en un 35 % respecto de modelos clásicos de dinámica molecular (medido mediante comparación con soluciones analíticas y validación cruzada).

Costos computacionales

La ejecución cuántica reduce los recursos computacionales necesarios, especialmente en problemas de alta complejidad. Si bien el acceso a hardware cuántico real aún presenta restricciones y costos elevados, los simuladores híbridos permiten disminuir el uso de recursos hasta en un 42 % comparados con simulaciones clásicas en problemas de optimización combinatoria. Esto se traduce en una reducción proyectada del 28 % en consumo energético y del 21 % en costos de infraestructura durante la fase de desarrollo.

7. Reproductibilidad y disponibilidad de artefactos

Con el objetivo de garantizar transparencia metodológica y replicabilidad experimental, se ha desarrollado y publicado un repositorio técnico que contiene la implementación completa del marco QSDLC propuesto en este trabajo. El repositorio integra la especificación formal de los tipos abstractos de datos cuánticos (Quantum Abstract Data Types, Q-ADT), las implementaciones algorítmicas utilizadas en la validación empírica y los scripts necesarios para reproducir los experimentos reportados.

La arquitectura del repositorio sigue un diseño modular que desacopla tres niveles fundamentales: (i) modelado conceptual basado en Q-UML, (ii) encapsulación estructural mediante Q-ADT y (iii) ejecución sobre plataformas cuánticas específicas. Esta separación permite una trazabilidad directa entre los artefactos conceptuales y su implementación ejecutable, reduciendo ambigüedades interpretativas y facilitando auditorías técnicas. Con el fin de evaluar la interoperabilidad tecnológica, el framework fue implementado sobre múltiples entornos de desarrollo cuántico, incluyendo Qiskit, Cirq y Q#. La abstracción de backend permite ejecutar los mismos artefactos Q-ADT en diferentes simuladores y, cuando es posible, en hardware cuántico real, asegurando independencia de proveedor y reduciendo riesgos de bloqueo tecnológico.

El repositorio incluye:

- Implementaciones completas de los algoritmos de Grover (Grover's algorithm), de optimización aproximada cuántica (Quantum Approximate Optimization Algorithm, QAOA) y del solucionador cuántico variacional de autovalores (Variational Quantum Eigensolver, VQE).
- Scripts de replicación con semillas determinísticas.
- Versionado explícito de dependencias y configuraciones de entorno. Pruebas unitarias automatizadas para validar la consistencia funcional del Q-ADT. Documentación técnica detallada y protocolo de ejecución experimental.

Asimismo, el repositorio ha sido versionado y archivado con un identificador persistente (DOI), permitiendo su citación formal y asegurando su disponibilidad a largo plazo. Su estructura y documentación cumplen con los principios FAIR (findable, accessible, interoperable, reusable), favoreciendo la reutilización científica y la evaluación independiente de los resultados.

Esta infraestructura de soporte no solo fortalece la validez empírica del estudio, sino que consolida el QSDLC como un marco metodológico reproducible, extensible y tecnológicamente interoperable dentro del ecosistema de ingeniería de software cuántico.

8. Conclusiones

Este estudio propuso un ciclo de vida de desarrollo de software cuántico (Quantum Software Development Life Cycle, QSDLC) como marco metodológico estructurado para el desarrollo de aplicaciones en entornos híbridos clásico-cuánticos. A diferencia de enfoques

centrados exclusivamente en herramientas de programación, el QSDLC integra fases tradicionales de la ingeniería de software con etapas específicas orientadas a la validación probabilística, la mitigación de ruido y decoherencia, y el mantenimiento adaptativo en sistemas no deterministas.

Este trabajo contribuye mediante: (i) la formalización de un ciclo de vida específico para software cuántico que articula prácticas clásicas con requerimientos físicos propios de la computación cuántica; (ii) la introducción de artefactos conceptuales originales —historias de usuario cuánticas, Q-UML y Q-ADT— que fortalecen la trazabilidad, la modularidad y la estandarización del diseño híbrido; y (iii) la validación empírica del modelo mediante la implementación de algoritmos representativos como el algoritmo de Grover (Grover's algorithm), el algoritmo de optimización aproximada cuántica (Quantum Approximate Optimization Algorithm) y el solucionador cuántico variacional de autovalores (Variational Quantum Eigensolver).

Los resultados experimentales evidenciaron reducciones significativas en el tiempo de ejecución y mejoras en la eficiencia energética frente a enfoques clásicos equivalentes, aunque acompañadas de mayores tasas de error inherentes a las limitaciones actuales del hardware cuántico. Estos hallazgos confirman que la incorporación de fases especializadas dentro del ciclo de vida contribuye a la sistematización, la reproducibilidad y la gobernanza del desarrollo cuántico, elementos esenciales para su futura estandarización.

Desde una perspectiva práctica, el QSDLC ofrece una guía reproducible para integrar algoritmos cuánticos en flujos híbridos, incrementando la productividad y la calidad del software. En el plano teórico, el modelo fortalece la consolidación de la ingeniería de software cuántico (Quantum Software Engineering, QSE) como disciplina emergente, al proporcionar un lenguaje común que reduce la brecha entre la ingeniería de software y la física cuántica.

Entre las principales limitaciones se identifican la dependencia de simuladores, la validación restringida a dispositivos de menos de 100 qubits y la ausencia de integración formal con marcos de gestión de riesgos híbridos. Futuros trabajos deberán extender la validación a hardware de gran escala, promover estándares internacionales —potencialmente bajo organismos como IEEE e ISO— e integrar el modelo en entornos CI/CD automatizados.

En conjunto, el QSDLC establece una base metodológica sólida para acompañar la transición desde la experimentación cuántica hacia prácticas industriales estandarizadas.

Rol del autor

- **Fabián Lizardo Caicedo Goyes:** investigación, análisis formal, escritura – borrador original, escritura – revisión y edición.

Referencias

[1] A. Khan, D. Taibi, C. M. Perrault, and M. A. Akbar, "Advancing quantum software engineering: A vision of hybrid full-stack iterative model," in *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '25. ACM, Mar. 2025, pp. 1444–1448. [Online]. Available: https://doi.org/10.1145/3672608.3707725

[2] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, "Quantum chemistry in the age of quantum computing," *Chemical Reviews*, vol. 119, no. 19, pp. 10 856–10 915, Aug. 2019. [Online]. Available: https://doi.org/10.1021/acs.chemrev.8b00803

[3] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, no. 1, Jan. 2016. [Online]. Available: https://doi.org/10.1038/npjqi.2015.23

[4] N. D. Mermin, *Quantum Computer Science: An Introduction*. Cambridge, UK: Cambridge University Press, 2007. [Online]. Available: https://doi.org/10.1017/CBO9780511813870

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. Cambridge, UK: Cambridge University Press, 2010. [Online]. Available: https://doi.org/10.1017/CBO9780511976667

[6] D. McMahon, *Quantum Computing Explained*. Hoboken, NJ, USA: Wiley, 2007. [Online]. Available: https://doi.org/10.1002/9780470186742

[7] T. Muske and A. Serebrenik, "Survey of approaches for postprocessing of static analysis alarms," *ACM Computing Surveys*, vol. 55, no. 3, pp. 1–39, Feb. 2022. [Online]. Available: https://doi.org/10.1145/3494521

[8] R. LaRose, "Overview and comparison of gate level quantum software platforms," *Quantum*, vol. 3, p. 130, 2019. [Online]. Available: https://doi.org/10.22331/q-2019-03-25-130

[9] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. de Oliveira Filho, M. Papendrecht, J. Rabbie, F. Rozpędek, M. Skrzypczyk, L. Wubben, W. de Jong,

D. Podareanu, A. Torres-Knoop, D. Elkouss, and S. Wehner, "Netsquid, a network simulator for quantum information using discrete events," *Communications Physics*, vol. 4, no. 1, 2021. [Online]. Available: https://doi.org/10.1038/s42005-021-00647-8

[10] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[11] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, "Hybrid quantum-classical algorithms and quantum error mitigation," *Journal of the Physical Society of Japan*, vol. 90, no. 3, p. 032001, Mar. 2021. [Online]. Available: https://doi.org/10.7566/JPSJ.90.032001

[12] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1666-5

[13] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest, "Measurement-based quantum computation," *Nature Physics*, vol. 5, no. 1, pp. 19–26, Jan. 2009. [Online]. Available: https://doi.org/10.1038/nphys1157

[14] M. Piattini, G. Peterssen Nodarse, R. Pérez-Castillo, J. L. Hevia Oliver, M. Serrano, G. Hernández González, I. Guzmán, C. Andrés Paradela, M. Polo, E. Murina, L. Jiménez Navajas, J. Marqueño, R. Gallego, J. Tura, F. Phillipson, J. Murillo, A. Niño, and M. Rodríguez, "The talavera manifesto for quantum software engineering and programming," *Software Quality Journal*, 03 2020. [Online]. Available: https://upsalesiana.ec/ing36ar4r2

[15] J. Zhao, "Quantum software engineering: Landscapes and horizons," *arXiv*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2007.07047

[16] B. Weder, J. Barzen, F. Leymann, M. Salm, and D. Vietz, "The quantum software lifecycle," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software*, ser. ESEC/FSE '20. ACM, Nov. 2020, pp. 2–9. [Online]. Available: https://doi.org/10.1145/3412451.3428497

[17] E. Desdentado, C. Calero, M. A. Moraga, and F. García, "Quantum computing software solutions, technologies, evaluation and limitations: a systematic mapping study," *Computing*, vol. 107, no. 5, Apr. 2025. [Online]. Available: https://doi.org/10.1007/s00607-025-01459-2

[18] A. Strikis, D. Qin, Y. Chen, S. C. Benjamin, and Y. Li, "Learning-based quantum error mitigation," *PRX Quantum*, vol. 2, no. 4, p. 040330, Nov. 2021. [Online]. Available: https://doi.org/10.1103/PRXQuantum.2.040330

[19] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Review X*, vol. 10, no. 2, p. 021067, 2020. [Online]. Available: https://doi.org/10.1103/PhysRevX.10.021067

[20] M. Fingerhuth, T. Babej, and P. Wittek, "Open source software in quantum computing," *PLOS ONE*, vol. 13, no. 12, p. e0208561, Dec. 2018. [Online]. Available: https://doi.org/10.1371/journal.pone.0208561

[21] M. Mohseni, P. Read, H. Neven, S. Boixo, V. Denchev, R. Babbush, A. Fowler, V. Smelyanskiy, and J. Martinis, "Commercialize quantum technologies in five years," *Nature*, vol. 543, no. 7644, pp. 171–174, Mar. 2017. [Online]. Available: https://doi.org/10.1038/543171a

[22] E. Kurian, D. Briola, P. Braione, and G. Denaro, "Automatically generating test cases for safety-critical software via symbolic execution," *Journal of Systems and Software*, vol. 199, p. 111629, May 2023. [Online]. Available: https://doi.org/10.1016/j.jss.2023.111629

[23] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K.

Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, "Noisy intermediate-scale quantum algorithms," *Reviews of Modern Physics*, vol. 94, no. 1, p. 015004, Feb. 2022. [Online]. Available: https://doi.org/10.1103/RevModPhys.94.015004

[24] W. H. Zurek, "Decoherence, einselection, and the quantum origins of the classical," *Reviews of Modern Physics*, vol. 75, no. 3, pp. 715–775, May 2003. [Online]. Available: http://doi.org/10.1103/RevModPhys.75.715

[25] D. J. Bernstein, *Introduction to post-quantum cryptography.* Springer Berlin Heidelberg, pp. 1–14. [Online]. Available: https://doi.org/10.1007/978-3-540-88702-7_1

[26] S.-H. Hung, K. Hietala, S. Zhu, M. Ying, M. Hicks, and X. Wu, "Quantitative robustness analysis of quantum programs," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–29, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3290344

[27] C. A. Pérez-Delgado, *A Quantum Software Modeling Language.* Springer International Publishing, 2022, pp. 103–119. [Online]. Available: https://doi.org/10.1007/978-3-031-05324-5_6

[28] S.-J. Ran, "Encoding of matrix product states into quantum circuits of one- and two-qubit gates," *Physical Review A*, vol. 101, no. 3, p. 032310, Mar. 2020. [Online]. Available: https://doi.org/10.1103/PhysRevA.101.032310

[29] M. Piattini, M. Serrano, R. Perez-Castillo, G. Petersen, and J. L. Hevia, "Toward a quantum software engineering," *IT Professional*, vol. 23, no. 1, pp. 62–66, Jan. 2021. [Online]. Available: https://doi.org/10.1109/mitp.2020.3019522

[30] V. M. Mostofi, D. Krishnamurthy, and M. Arlitt, "Fast and efficient performance tuning of microservices," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD).* IEEE, 2021, pp. 515–520. [Online]. Available: https://doi.org/10.1109/CLOUD53861.2021.00067

[31] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. Upper Saddle River, NJ, USA: Pearson, 2011. [Online]. Available: https://upsalesiana.ec/ing36ar4r11

[32] A. Whitmill, S. Kim, V. Rojas, F. Gulraiz, K. Afreen, M. Jain, M. Singh, and I.-W. Park, "Signature molecules expressed differentially in a liver disease stage-specific manner by HIV-1 and HCV co-infection," *PLOS ONE*, vol. 13, no. 8, p. e0202524, Aug. 2018. [Online]. Available: https://doi.org/10.1371/journal.pone.0202524

[33] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean, "Quantum advantage in learning from experiments," *Science*, vol. 376, no. 6598, pp. 1182–1186, 2022. [Online]. Available: https://doi.org/10.1126/science.abn7293