Modelo de Características Extendido para Sistemas Híbridos (Cuántico-Clásicos)

Samuel Sepúlveda[1], Ricardo Pérez-Castillo[2], Mario Piattini[3]

[1]Departamento de Ciencias de la Computación e Informática – Universidad de La Frontera – Av. Francisco Salazar 01145 – Temuco – 4811230 – Chile

[2]Facultad de Ciencias Sociales y Tecnologías de la Información – Universidad de Castilla-La Mancha – Av. Real Fábrica de Sedas s/n Talavera de la Reina – 45600 – España

[3]Departamento de Tecnologías y Sistemas de Información – Universidad de Castilla-La Mancha – Paseo de la Universidad 4 – Ciudad Real – 13071 – España

samuel.sepulveda@ufrontera.cl

ricardo.pdelcastillo,mario.piattini@uclm.es

Abstract. *Quantum computing (QC) and quantum software engineering (QSE) face challenges due to the diversity of algorithms, languages, and integration strategies. This paper proposes a quantum-classical hybrid software development model following a Software Product Lines approach. Based on a previous model and a literature search, essential components were identified and structured in a hierarchy to represent the variability in these systems. The resulting model facilitates variability and configuration management of hybrid architectures, setting the basis for further evaluation and refinement and contributing to a more effective methodology for integrating QC and QSE.*

Resumen. *La computación cuántica (CQ) y la ingeniería de software cuántico (ISC), enfrentan desafíos debido a la diversidad de algoritmos, lenguajes y estrategias de integración. Este trabajo propone un modelo para el desarrollo de software híbrido cuántico-clásico, siguiendo un enfoque de Líneas de Producto de Software. A partir de un modelo previo y una búsqueda de la literatura, se identificaron componentes esenciales y se estructuraron en una jerarquía para representar la variabilidad en estos sistemas. El modelo resultante facilita la gestión de variabilidad y configuración de arquitecturas híbridas, estableciendo las bases para futuras evaluaciones y refinamientos, contribuyendo a una metodología más efectiva para la integración de CQ e ISC.*

1. Introducción

En el ámbito de la computación cuántica (CQ) y la ingeniería de software cuántico (ISC), la multiplicidad de algoritmos, lenguajes de programación, mecanismos de integración genera importantes desafíos al momento de adoptar y adaptar soluciones cuánticas [Salam and Ilyas 2024]. La falta de pautas que guíen la adopción de estas tecnologías, en continua evolución, a menudo dificulta su integración efectiva [Bisicchia et al. 2024]. Además, los sistemas software del futuro no serán puramente cuánticos, sino que estos serán sistemas híbridos (clásico-cuánticos) donde parte de la funcionalidad será soportada

por software cuántico que deberá ser integrado en los sistemas software clásicos heredados [Pérez-Castillo et al. 2021].

A pesar de los avances significativos en ISC y Líneas de Productos de Software (LPS), no se han desarrollado propuestas completamente integradas que faciliten la transición de grandes sistemas clásicos hacia sistemas híbridos con alta variabilidad [Carbonelli et al. 2024]. Si bien existen propuestas que exploran el uso de LPS y modelos de características (MC) para abordar ciertos aspectos de esta transición [Sepúlveda et al. 2024, Sepúlveda et al. 2025], dichas propuestas no abordan de manera integral los desafíos asociados con la gestión de variabilidad y la escalabilidad en sistemas híbridos.

La presente investigación busca consolidar y expandir estas ideas mediante el desarrollo de un modelo más robusto, que integre ISC y LPS de manera eficiente, gestionando la variabilidad en diversos escenarios y bajo distintas restricciones. Al organizar y clasificar la compleja variedad de tecnologías cuánticas, se busca facilitar una comprensión más clara e identificar de manera sistemática las características relevantes según sus relaciones intrínsecas. En este contexto, se busca responder la siguiente pregunta: ¿Cómo la adaptación de metodologías tradicionales con artefactos cuánticos puede facilitar una integración entre sistemas clásicos y cuánticos al usar conceptos de LPS? Para abordar esta cuestión, se plantea la hipótesis de que la adaptación de metodologías tradicionales, complementadas con un artefacto diseñado para incorporar principios cuánticos, permitirá un diseño y una implementación con más calidad de los sistemas híbridos.

El resto del artículo se estructura como sigue. La sección 2 presenta los conceptos fundamentales relacionados con la propuesta. Las secciones 3 y 4 presentan el desarrollo de la propuesta y una discusión de los principales resultados, respectivamente. Finalmente, la sección 5 presenta las conclusiones del trabajo realizado.

2. Antecedentes generales

Esta sección presenta los conceptos básicos de la QC (Sección 2.1), seguido de la ISC (Sección 2.2), y finalmente los conceptos de LPS y la variabilidad (Sección 2.3).

2.1. Conceptos básicos

La CQ utiliza los principios de la superposición y el entrelazamiento para procesar información de manera más eficiente que las computadoras clásicas en ciertos tipos de problemas [Nielsen and Chuang 2001].

A diferencia de los bits clásicos (0 y 1), los qubits manejan ambos estados simultáneamente de forma probabilística, lo que permite realizar cálculos paralelos con mayor eficiencia. Este paradigma ha demostrado su potencial en áreas como la optimización, criptografía y simulación molecular. Aunque las tecnologías actuales se limitan a sistemas NISQ (Noisy Intermediate-Scale Quantum), se espera que su desarrollo impacte significativamente la industria del software según se consiguen dispositivos FTQC (Fault-tolerant Quantum Computing) tolerantes a fallos [Chen and Rengaswamy 2024].

2.2. Ingeniería de software cuántica

La ISC es una disciplina emergente que busca adaptar metodologías tradicionales de software para gestionar la complejidad del desarrollo cuántico. El concepto de

ISC [Piattini et al. 2020, Zhao 2021, Serrano et al. 2022] ha emergido en los últimos años como un conjunto de procesos, métodos y herramientas diseñados para maximizar el potencial de los sistemas de computación cuántica. El Software Engineering Institute [Carleton et al. 2021] menciona desafíos como la integración de algoritmos cuánticos en sistemas híbridos y la falta de herramientas formales para gestionar la incertidumbre en sistemas cuánticos.

Por otro lado, [Exman et al. 2024] presenta el estado actual de la ISC y el diseño de sistemas cuánticos, independientemente del hardware específico. Los autores abordan aspectos teóricos del software cuántico y aplica conceptos clásicos de ingeniería de software, como enfoques ágiles de desarrollo, validación, medición y despliegue, en entornos cuánticos o híbridos. Además, se complementa con diversas aplicaciones industriales. Un estudio secundario muestra que la mayor parte de la investigación en ISC se ha centrado en pruebas de software y en el uso de frameworks como Qiskit [De Stefano et al. 2024]. Sin embargo, todavía queda un vacío importante en áreas de gestión y diseño a nivel arquitectónico para sistemas híbridos.

Si bien ya se ha llevado a cabo un trabajo preliminar que aborda el proceso basado en principios de LPS y el uso de MC para software híbrido, la propuesta actual se centra en abarcar los diferentes aspectos involucrados en el desarrollo de software híbrido e integrar las funcionalidades necesarias para gestionar su variabilidad de manera integral.

2.3. Líneas de Productos de Software (LPS) y Modelado de Variabilidad

Las LPS se han consolidado como un enfoque eficaz para gestionar la variabilidad en sistemas complejos mediante modelos de características (MC). Estos modelos representan la diversidad de productos posibles en una línea de software, permitiendo generar múltiples configuraciones desde una base común. Benavides et al. (2010) destacan que los MC son esenciales para gestionar la complejidad en LPS, aunque aún existen retos en su aplicación a entornos colaborativos y dinámicos [Benavides et al. 2010].

Galindo et al. (2019) examinan cómo los MC han sido adoptados en la industria y sugieren que la combinación de LPS con tecnologías emergentes, como la computación cuántica, podría aumentar significativamente su aplicabilidad [Galindo et al. 2019]. Además, estudios recientes han explorado nuevas áreas de aplicación, como los sistemas ciberfísicos y los vehículos autónomos submarinos, lo que demuestra la flexibilidad de las LPS para adaptarse a dominios emergentes [Safdar et al. 2016, Cares et al. 2022].

3. Modelo de Características para Software Híbrido

La propuesta de investigación principal es un modelo de variabilidad para el desarrollo de software híbrido. Aunque un MC preliminar ya ha sido publicado por [Sepúlveda et al. 2024, Sepúlveda et al. 2025], estos trabajos exploran cómo puede usarse el MC propuesto en el contexto general de LPS en el desarrollo de software híbrido. No obstante, dicho MC no resulta ser completo ni preciso como para cubrir toda la casuística en este tipo de desarrollos de manera eficiente, gestionando la variabilidad en diversos escenarios y bajo distintas restricciones. Este es precisamente el objetivo del presente artículo. Para ello, se ha llevado a cabo una búsqueda en la literatura para descubrir y clasificar los componentes esenciales de la CQ —por ejemplo, algoritmos, lenguajes, mecanismos de integración y métodos de corrección de errores— junto con sus propiedades

definitorias. Los hallazgos fueron discutidos en cuanto a su importancia y relevancia entre los autores, desarrollando un modelo cuya estructura jerárquica encapsula la diversidad de los elementos y restricciones para el software híbrido.

El modelo propuesto se define mediante formato textual con UVL (*Universal Variability Language*) [Benavides et al. 2024, Fadhlillah and Rabiser 2024]. Definir un modelo de variabilidad en UVL en lugar de un árbol gráfico facilita la escalabilidad, edición y mantenimiento en texto plano, permitiendo integración con herramientas de análisis y automatización [Rabiser 2024]. Además, UVL es más flexible para expresar restricciones como opcionalidad, alternativas y dependencias, evitando la complejidad visual de los diagramas cuando el modelo crece. Para complementar el presente estudio, se recomienda consultar el material proporcionado por [Sepúlveda et al. 2025], donde se detalla la representación gráfica del MC y se incluye su archivo de configuración en formato JSON.

La estructura del modelo se presenta en el Código 1. En las siguientes subsecciones se describen los principales grupos de características que se consideran: el tipo de funcionalidad soportada, tipo de algoritmo, los framework y lenguajes de programación, el modelo de integración, y restricciones software basadas en el hardware.

```
HQC_SPL

features:
    @Functionality:
        ...
    @Algorithm:
        ...
    @Programming:
        ...
    @Integration_model:
        ...
    @Quantum_HW_constraint:
        ...
# restricciones para todos los elmentos del modelo
constraints:
    ...
```

Código 1. Modelo de características en UVL para software híbrido

3.1. Tipo de funcionalidad soportada

Un sistema de software híbrido cuántico-clásico integra componentes cuánticos y clásicos para aprovechar las fortalezas de ambos paradigmas [Faro et al. 2023]. En la propia fase de análisis de requisitos [Pendenti et al. 2024], una de las primeras preguntas es si una determinada funcionalidad deberá ser soportada por software clásico o cuántico (véase el Código 2). Existen algunas aproximaciones para determinar si una determinada funcionalidad debe ser soportada por software clásico o cuántico. Por ejemplo, [Chancellor et al. 2020] proporcionan un marco de toma de decisiones para casos de uso de algoritmos cuánticos y cuántico-inspirados. Obviamente, esta decisión estará condicionada también por otras restricciones (entre ellas de hardware) que son modeladas en otro subárbol de características del MC propuesto. De forma similar, el propio algoritmo escogido condicionará la elección, así, por ejemplo, si se pretende resolver un problema

mediante *machine learning*, también tendrá que decidirse si se hace mediante software clásico o *quantum machine learning* (QML) [Nadim et al. 2024].

Adicionalmente, es posible distinguir dos principales paradigmas de computación cuántica (véase el Código 2): la computación cuántica basada en puertas y la computación cuántica adiabática, también conocida en su implementación práctica como *Quantum Annealing* (QA) [McGeoch 2020]. Por un lado, la computación basada en puertas también se conoce como computación cuántica universal, ya que, en teoría, puede abordar todos los problemas objetivo (dado que es Turing-completa), mientras que los sistemas de QA se utilizan exclusivamente para problemas de optimización y han logrado cierta penetración en el mercado, como es el caso de D-Wave [Kahn 2020].

```
1    @Functionality:
2        Classical
3        @Quantum:
4            Gate_based
5            Adiabatic
```

Código 2. Características definidas en UVL para el tipo de funcionalidad software

3.2. Algoritmos Software

Los algoritmos híbridos como el Variational Quantum Eigensolver (VQE) y el Quantum Approximate Optimization Algorithm (QAOA) combinan el procesamiento cuántico y clásico. Estos algoritmos ejecutan subrutinas cuánticas, cuyas salidas son utilizadas por procesos clásicos para optimizar los resultados en tiempo real. La iteración continua entre ambas partes es clave, lo que permite que las decisiones clásicas condicionen operaciones cuánticas futuras dentro de un mismo ciclo de ejecución.

El trabajo de [Qi et al. 2024] presenta una revisión y aplicación de los algoritmos variacionales, así como sus aplicaciones y desafíos asociados su implementación dispositivos cuánticos actuales. Una revisión específica de los algoritmo VQE y QAOA, sus implementaciones prácticas y varios resultados experimentales se muestran en [Grange et al. 2023]. La variabilidad de algoritmos que se debe manejar es, en primer lugar, si estos son clásicos, cuánticos o híbridos.

En primer lugar, los algoritmos clásicos utilizan procesamiento determinista basado en bits clásicos y no aprovechan, de ninguna forma, fenómenos cuánticos como superposición o entrelazamiento. Estos son altamente maduros y aplicables a una amplia gama de problemas. Por ello, en este artículo se obvia un listado pormenorizado de estos algoritmos, aunque estos se pueden clasificar y organizar atendiendo a diferentes criterios:

- Propósito principal: Procesamiento de datos (clasificación, búsqueda, ordenación); Optimización (combinatoria, estocástica); Inteligencia artificial (aprendizaje supervisado, no supervisado, por refuerzo), Criptografía y seguridad (cifrado simétrico, cifrado asimétrico, hashing).
- Complejidad y enfoque: Deterministas (divide y vencerás, voraces, dinámicos); No deterministas (probabilísticos, heurísticos).
- Paradigma de programación: Iterativos, Recursivos, Paralelos, Distribuidos.
- Metodología de desarrollo: Exactos, Aproximados, Heurísticos.

Por otra parte, los desafíos de la gestión de la variabilidad en sistemas software híbridos residen en los algoritmos cuánticos puros o híbridos.

Los algoritmos cuánticos aprovechan fenómenos cuánticos como superposición, entrelazamiento e interferencia y están especialmente diseñados para aprovechar la aceleración cuántica en problemas específicos. A continuación se presenta una clasificación de éstos que divide en dos grandes grupos: FTQC y NISQ [Ezratty 2024]. A parte, se consideran algoritmos *quantum annealing* específicos para computación cuántica adiabática.

FTQC (Fault-Tolerant Quantum Computing): Son algoritmos diseñados para computadoras cuánticas tolerantes a fallos, que corrigen errores durante la ejecución. Se centran en aplicaciones de gran escala, como la simulación precisa de sistemas físicos y criptografía, con aceleración garantizada a largo plazo.

- Deutsch-Jozsa: Verificar si una función es constante o balanceada, consigue una ventaja exponencial ($O(1)$).
- Bernstein-Vazirani: Descubrir una clave secreta codificada en una función matemática en una sola consulta, consigue una ventaja exponencial ($O(1)$).
- Grover: Búsqueda en una base de datos no estructurada, consigue una ventaja cuadrática ($O(\sqrt{n})$).
- Simon: Encontrar un patrón oculto en una función matemática, consigue una ventaja exponencial ($O(1)$).
- Shor: Factorización de números semi-primos. Ventaja Exponencial para enteros grandes.
- QFT (Quantum Fourier Transform): Convertir un conjunto de números en una representación de frecuencias, consigue una ventaja exponencial ($O(1/\epsilon^2)$).
- QPE (Quantum Phase Estimation): Permite encontrar la fase asociada a un operador cuántico, consigue una ventaja exponencial ($O(1/\epsilon)$).
- HHL (Harrows-Hassidim-Lloyd): Resolución de sistemas lineales. Ventaja exponencial en matrices dispersas.

NISQ (Noisy Intermediate-Scale Quantum): Son algoritmos para dispositivos cuánticos actuales con ruido, sin corrección de errores completa. Son heurísticos y se usan en problemas como optimización y *machine learning*, pero su rendimiento no está garantizado de manera general.

- QAOA (Quantum Approximate Optimization Algorithm): Optimización combinatoria desde una función objetivo codificada en ansatz.
- VQE (Variational Quantum Eigensolver): Obtención de energías de estado base.
- QML (Quantum Machine Learning): Clasificación y clustering basado en QNN (Quantum Neural Networks). Potencial mejora en problemas de alta dimensionalidad.

Finalmente, los algoritmos híbridos combinan procesamiento clásico y cuántico. Usan computadoras clásicas para manejar preprocesamiento o integración con datos, delegando tareas específicas al hardware cuántico.

- Quantum-Assisted Annealing: Optimización combinatoria que mezcla simulaciones clásicas y cálculos cuánticos.
- Clustering híbrido, realiza el preprocesamiento clásico para el particionado en problemas más manejables y luego aplica un algoritmo cuántico.

- QAOA clásico-cuántico: Preprocesamiento clásico para reducir restricciones complejas y optimización cuántica del objetivo.

A modo de resumen, el Código 3 muestra el subárbol UVL que modela las características para el tipo de algoritmo considerado.

```
@Algorithm:
    {Quantum, Hybrid, Classical}:
    @Quantum:
        {0..1} {FTQC, NISQ, Quantum_Annealing}:
            FTQC:
                Deutsch_Jozsa
                Bernstein-Varizani
                Grover
                Simon
                Shor
                QFT
                QPE
                HHL
            NISQ:
                QAOA
                VQE
                QML
            Quantum_Annealing
    @Hybrid:
        Quantum_Assisted_Annealing
        Hybrid_Clustering
        Classical_Quantum_QAOA
    @Classical
```

Código 3. Características definidas en UVL para el tipo de algoritmo

3.3. Lenguajes de Programación

Los marcos de trabajo y lenguajes de programación cuánticos han evolucionado significativamente para facilitar el desarrollo de software híbrido cuántico-clásico, presentando diferencias clave en compatibilidad, sintaxis y optimización. Un análisis comparativo de plataformas como Qiskit, Cirq, Forest y Q# destaca variaciones en requisitos de instalación, sintaxis y capacidades de simulación, lo que influye en la elección de herramientas según el caso de uso específico [LaRose 2019].

La diversidad de lenguajes cuánticos permite adaptaciones para diferentes paradigmas de computación, siendo algunos más adecuados para modelos de circuitos cuánticos y otros para enfoques adiabáticos o de simulación [Heim et al. 2020]. Además, la selección del compilador cuántico adecuado es crucial, ya que impacta en la optimización de circuitos y en la eficiencia de ejecución en hardware real, lo que resalta la importancia de considerar estos factores en arquitecturas híbridas [Salm et al. 2021].

Finalmente, la portabilidad entre diferentes marcos sigue siendo un desafío, pues las diferencias en especificaciones dificultan la interoperabilidad y la reutilización de código cuántico en distintas plataformas [Schönberger et al. 2022]. Estos estudios respaldan la necesidad de estructurar de manera modular los elementos de programación en modelos como UVL, asegurando compatibilidad, flexibilidad y eficiencia en el desarrollo de software híbrido cuántico-clásico.

A modo de resumen, el Código 4 muestra el sub árbol UVL que modela las características para el tipo de framework y lenguaje de programación considerado.

```
@Programming:
    {Quantum_framework, Language}:
    @Quantum_framework:
        Qiskit
        dWave
        Cirq
        Braket SDK
    @Language:
        {quantum, classical}:
        @quantum:
            QASM
            QSharp
        @classical:
            Python
            CSharp
```

Código 4. Características definidas en UVL para el tipo de funcionalidad programación

3.4. Modelos de integración software

La integración entre computación clásica y cuántica es un desafío fundamental en la adopción de tecnologías cuánticas dentro de sistemas informáticos convencionales. Existen diversos enfoques para lograr esta interoperabilidad basada en servicios [Ahmad et al. 2024], los cuales se pueden agrupar en tres modelos principales (véase el Código 5): integración mediante APIs y computación en la nube, sistemas Quantum-SOA y orquestación mediante microservicios.

```
@Integration_model:
    {1..1} {Middleware_API, Quantum_SOA, Microservices}
```

Código 5. Características definidas en UVL para los tipos de integración software

El primer modelo, **integración mediante APIs y middleware en nube**, se basa en la interacción entre código clásico y cuántico a través de interfaces de programación de aplicaciones (APIs) y software development kits (SDKs) proporcionados por plataformas de computación cuántica en la nube. Frameworks como Qiskit, Cirq y Braket SDK permiten que lenguajes clásicos, como Python, interactúen con hardware cuántico remoto, mientras que esos servicios ofrecen infraestructura para ejecutar algoritmos cuánticos en sistemas distribuidos. Este enfoque facilita la adopción de la computación cuántica sin necesidad de poseer hardware especializado, además de permitir la integración de algoritmos cuánticos en flujos de trabajo convencionales. Sin embargo, la latencia en la comunicación y los costos asociados al uso de recursos cuánticos en la nube pueden representar limitaciones en ciertos escenarios.

El segundo modelo, **sistemas Quantum-SOA (Service-Oriented Architecture)**, aplica la arquitectura orientada a servicios para desacoplar el procesamiento cuántico y clásico en módulos independientes, permitiendo su integración flexible dentro de infraestructuras empresariales o científicas [Hevia et al. 2024]. En este enfoque, los sistemas cuánticos se exponen como servicios que pueden ser invocados bajo demanda a través de

protocolos estándar como REST. Por ejemplo, un sistema de optimización en la industria logística puede delegar cálculos NP-duros a un servicio cuántico, o bien, una plataforma de banca puede integrar módulos de encriptación cuántica como parte de su infraestructura de ciberseguridad. La principal ventaja de este modelo radica en la escalabilidad y reutilización de componentes, permitiendo que distintos sistemas clásicos accedan a las capacidades cuánticas sin modificar su arquitectura base. No obstante, la orquestación eficiente entre servicios cuánticos y clásicos sigue siendo un desafío, especialmente en entornos con restricciones de tiempo real. Un estudio secundario sobre la QC como servicio (QCaaS) se presenta en [Ahmad et al. 2023].

El tercer modelo, **orquestación mediante microservicios**, extiende la idea de modularidad introducida por Quantum-SOA al ámbito de la computación distribuida, organizando los distintos componentes clásicos y cuánticos en una arquitectura de microservicios [Garcia-Alonso et al. 2021]. En este enfoque, cada unidad funcional del sistema se encapsula en un microservicio independiente que puede comunicarse con otros módulos a través de una infraestructura de mensajería asincrónica. Un caso de uso típico es la integración de modelos de aprendizaje automático donde los preprocesadores clásicos envían datos a microservicios cuánticos especializados en clustering o reducción de dimensionalidad. Tecnologías como *Kubernetes* permiten gestionar contenedores cuánticos y clásicos en la misma infraestructura, facilitando la implementación de flujos de trabajo híbridos. Aunque este modelo ofrece una gran flexibilidad y escalabilidad, la gestión de latencias y la sincronización de flujos de datos entre sistemas clásicos y cuánticos pueden representar desafíos significativos en escenarios de alta concurrencia. Una implementación de microservicios cuánticos para resolver el problema del viajante, ventajas y limitaciones se presenta en [Rojo et al. 2021].

3.5. Restricciones software basadas en hardware cuántico

Dada la naturaleza intrínseca del software cuántico y el alto impacto que tiene el hardware cuántico en él (debido a la madurez actual de la tecnología), existen numerosas restricciones derivadas del hardware que impactan fuertemente en los diferentes productos software dependiendo de la variabilidad escogida. El Código 6 resume todos estos elementos, que se explican a continuación.

Los sistemas híbridos requieren la integración de un procesador clásico y un procesador cuántico (QPU), cuya comunicación es fundamental para tareas como el control del flujo del programa y la evaluación de resultados. Los registros clásicos gestionan instrucciones y mediciones de qubits, mientras que los registros cuánticos operan los estados necesarios para las operaciones del sistema. La construcción de estas arquitecturas presenta desafíos significativos [Stirbu and Mikkonen 2023], requiriendo la adaptación del software para satisfacer las necesidades específicas de los sistemas híbridos [Ahmad et al. 2022].

La mitigación de errores y la reducción de la latencia son esenciales para el rendimiento en sistemas híbridos cuántico-clásicos. Métodos avanzados para la corrección y mitigación de errores cuánticos han demostrado reducciones significativas en experimentos recientes [Cai et al. 2023, Czarnik et al. 2021], mientras que la integración eficiente entre unidades clásicas y cuánticas contribuye a minimizar tiempos de espera. Además, las redes de comunicación cuántica han mostrado ventajas clave en términos de eficiencia y seguridad, así como en la reducción de la latencia en redes de gran escala [Ferrara et al. 2021, K.S. and Gowda 2023, Sekavčnik and Nötzel 2022].

```
    @Quantum_HW_constraint:
        {Error correction,quality,optimization}:
        @Error correction:
            logic qubits
            dynamic correction
        @optimization:
            memory
            task schedule
        quality
```

Código 6. Características definidas en UVL para las restricciones basadas en hardware que impactan los diferentes productos software

3.6. Restricciones en el Modelo de Características

Además de las relaciones entre características, en el modelo de variabilidad se permiten restricciones cruzadas (*cross-tree constraints*). Estas restricciones pueden ser de dos tipos: «requires» y «excludes». La relación «requires» entre una característica *A* y una característica *B* indica que la selección de *A* obliga a seleccionar también *B*. Por otro lado, la relación «excludes» entre *A* y *B* significa que ambas características no pueden formar parte del mismo producto.

El Código 7 muestra el subárbol UVL que modela algunas de las restricciones entre características del modelo. Debido a las limitaciones de extensión del artículo, sólo se muestran cuatro de ellas.

```
constraints:
    # Restricción: Quantum require Quantum_HW_constraint
    Quantum -> Quantum_HW_constraint

    # Restricción: NISQ excluye Fault-Tolerant
    NISQ -> not Fault_Tolerant

    # Restricción: dWave require Annealing
    dWave -> Annealing

    # Restricción: Quantum_Algorithm requiere Functionality - Quantum
    Quantum_Algorithm -> Quantum
```

Código 7. Restricciones definidas en UVL entre algunas características del modelo

4. Discusión de Resultados

El modelo propuesto clarifica la interrelación entre tecnologías cuánticas, facilita comparaciones y destaca aspectos clave en sistemas híbridos cuántico-clásicos. Además, proporciona una base para futuras evaluaciones e integraciones, promoviendo una gestión más eficiente y adaptable de la variabilidad en el desarrollo de software cuántico.

4.1. Impacto esperado de las LPS y modelado de la variabilidad

La aplicación de LPS puede mejorar significativamente el diseño e implementación de sistemas híbridos cuántico-clásico en los elementos clave previamente mencionados.

En arquitecturas de procesamiento mixto, permite representar diversas configuraciones de procesadores cuánticos y clásicos, así como sus interacciones, facilitando la generación automática de arquitecturas adaptadas a distintos escenarios, como entornos locales o en la nube. Por ejemplo, se pueden definir variantes según si los QPUs están integrados localmente o accesibles a través de plataformas remotas.

Para los algoritmos híbridos, posibilita la gestión de variantes de algoritmos como VQE y QAOA, adaptando automáticamente componentes según las necesidades del sistema y las restricciones del hardware. A través de los MC, es factible modelar versiones y variantes paramétricas de los algoritmos, optimizando su configuración en tiempo real. Un ejemplo es la adaptación de VQE según la cantidad de qubits disponibles o el tiempo de coherencia del hardware cuántico.

En corrección de errores, se pueden representar múltiples estrategias y permitir la selección automática de la más adecuada según las condiciones del hardware. Una LPS facilita la integración de distintas técnicas en un sistema único, garantizando flexibilidad en su aplicación. Por ejemplo, se puede automatizar la elección entre corrección con qubits lógicos o mitigación basada en parámetros, considerando el tipo de QPU y el nivel de ruido en las mediciones.

4.2. Modelo propuesto

A continuación, se describen algunas de las diferencias clave entre el modelo propuesto y el modelo previo [Sepúlveda et al. 2025]. Se explican las modificaciones realizadas y cómo estas mejoran dicha propuesta.

Aspectos generales. Se introduce mayor granularidad, flexibilidad y precisión en la descripción de las características y restricciones. Esto facilita que el modelo sea más aplicable en escenarios reales, evitando inconsistencias y facilitando configuraciones válidas. Además, el enfoque actualizado refleja mejor las necesidades técnicas y prácticas de los sistemas híbridos, alineándose con los avances en ISC y LPS reportados en la literatura.

Uso de notación UVL. Ésta ofrece ventajas frente al modelado de la variabilidad usando MC, en particular usando modelos de texto. UVL permite una representación modular y estandarizada, facilitando la edición y mantenimiento sin depender de herramientas gráficas. Además, es compatible con herramientas de validación automática optimiza la gestión de restricciones y dependencias, asegurando configuraciones coherentes.

Interoperabilidad y algoritmos. Es posible combinar y optimizar algoritmos cuánticos y clásicos en flujos de trabajo híbridos, lo cual no estaba explícito en el modelo original. Este cambio refleja mejor la realidad actual de los sistemas híbridos cuántico-clásicos, donde algoritmos como VQE y QAOA requieren interacción continua con optimizadores clásicos. Esta nueva característica facilita el diseño de sistemas más eficientes y robustos.

Mejora en la Organización de HW-based Constraint. En este nuevo modelo, HW-based Constraint se reorganiza en subcaracterísticas como Calidad y Corrección de Errores, mejorando la granularidad del modelo. Esto permite una representación más detallada de las limitaciones y capacidades del hardware cuántico, asegurando que las restricciones se adapten mejor a las necesidades específicas de cada configuración.

Latencia y Comunicación como restricciones. Ahora estas propiedades se consideran como restricciones, lo cual simplifica el modelo al considerar que la latencia y la comunicación son propiedades del hardware y no funcionalidades seleccionables, reflejando con mayor precisión el impacto técnico de estas propiedades sin sobrecargar la descripción funcional. Además, este enfoque facilita su evaluación en configuraciones específicas sin introducir redundancia.

Posibilidad de extensión. Se considera agregar atributos a ciertas características del modelo, que permitan calcular métricas cuantitativas para evaluar restricciones como costos, tiempos de procesamiento, número de qubits requeridos, etc. Además, se considera extender el modelo para incluir nuevos algoritmos y arquitecturas emergentes. Por último, se considera incorporar características para sistemas distribuidos y auto adaptativos, incluyendo conceptos de LPS dinámicas.

5. Conclusiones

La propuesta presentada constituye un marco flexible y sólido para caracterizar soluciones en entornos híbridos cuántico-clásicos. Al incorporar conceptos y subdominios nucleares de la CQ dentro de una cuasi-taxonomía, el trabajo permite a investigadores y practicantes comprender los componentes esenciales y sus interconexiones, sentando las bases para una integración más coherente y efectiva de CQ-ISC. La inclusión de los conceptos de LPS contribuye a una gestión eficiente de la variabilidad y personalización, facilitando la selección automática de configuraciones óptimas y la integración de componentes en diferentes escenarios.

Como trabajo futuro, se plantea aplicar el modelo en distintos escenarios de integración cuántico-clásicos para validar su efectividad y flexibilidad, así como ampliar la propuesta incorporando técnicas de optimización formal que gestionen la complejidad de la computación cuántico-clásica, realizando un análisis cuantitativo de costos, recursos y otros factores, maximizando la fidelidad y garantizando el cumplimiento de restricciones.

Agradecimientos

Samuel Sepúlveda cuenta con el apoyo de ANID, Chile, FONDECYT de Iniciación en Investigación, Proyecto Nº 11240702.

Este trabajo ha sido soportado por los proyectos PID2022-137944NB-I00 (Proyecto SMOOTH) y PDC2022-133051-I00 (Proyecto QU-ASAP) financiados por MCIN/AEI/ 10.13039/501100011033 y por la "Unión Europea NextGenerationEU/FEDER". Adicionalmente, ha sido posible gracias a la ayuda UNION (2022-GRIN-34110) para la realización de proyectos de investigación aplicada, en el marco del Plan Propio de Investigación de la UCLM (cofinanciados con FEDER).

El trabajo cuenta con el apoyo del Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo (CYTED), a través de RIPAISC - Red Iberoamericana para el avance de la Ingeniería de Software Cuántico (525RT0174).

Referencias

Ahmad, A., Altamimi, A. B., and Aqib, J. (2024). A reference architecture for quantum computing as a service. *Journal of King Saud University - Computer and Information Sciences*, 36(6):102094.

Ahmad, A., Khan, A. A., Waseem, M., Fahmideh, M., and Mikkonen, T. (2022). Towards process centered architecting for quantum software systems. In *2022 IEEE international conference on quantum software (QSW)*, pages 26–31. IEEE.

Ahmad, A., Waseem, M., Liang, P., Fehmideh, M., Khan, A. A., Reichelt, D. G., and Mikkonen, T. (2023). Engineering software systems for quantum computing as a service: A mapping study. *arXiv preprint arXiv:2303.14713*.

Benavides, D., Segura, S., and Ruiz-Cortés, A. (2010). Automated analysis of feature models 20 years later: A literature review. *Information systems*, 35(6):615–636.

Benavides, D., Sundermann, C., Feichtinger, K., Galindo, J. A., Rabiser, R., and Thüm, T. (2024). Uvl: Feature modelling with the universal variability language. *SSRN*.

Bisicchia, G., Garcia-Alonso, J., Murillo, J. M., and Brogi, A. (2024). From quantum mechanics to quantum software engineering. *arXiv preprint arXiv:2404.19428*.

Cai, Z., Babbush, R., Benjamin, S. C., Endo, S., Huggins, W. J., Li, Y., McClean, J. R., and O'Brien, T. E. (2023). Quantum error mitigation. *Rev. Mod. Phys.*, 95:045005.

Carbonelli, C., Felderer, M., Jung, M., Lobe, E., Lochau, M., Luber, S., Mauerer, W., Ramler, R., Schaefer, I., and Schroth, C. (2024). Challenges for quantum software engineering: An industrial application scenario perspective. In *Quantum Software: Aspects of Theory and System Design*, pages 311–335. Springer.

Cares, C., Lühr, D., Mora, S., Navarro, C., Olivares, L., Sepúlveda, S., and Vidal, G. (2022). Architecting autonomous underwater vehicles by adapting software product lines. In *Conference on Integrated Computer Technologies in Mechanical Engineering–Synergetic Engineering*, pages 719–730. Springer.

Carleton, A. D., Harper, E., Robert, J. E., Klein, M. H., De Niz, D., Desautels, E., Goodenough, J. B., Holland, C., Ozkaya, I., and Schmidt, D. (2021). Architecting the future of software engineering: A national agenda for software engineering research and development. Report, Software Engineering Institute, Carnegie Mellon University.

Chancellor, N., Cumming, R., and Thomas, T. (2020). Toward a standardized methodology for constructing quantum computing use cases. `https://arxiv.org/abs/2006.05846v1`. ArXiv preprint.

Chen, Z. and Rengaswamy, N. (2024). Tailoring fault-tolerance to quantum algorithms.

Czarnik, P., Arrasmith, A., Coles, P. J., and Cincio, L. (2021). Error mitigation with Clifford quantum-circuit data. *Quantum*, 5:592.

De Stefano, M., Pecorelli, F., Di Nucci, D., Palomba, F., and De Lucia, A. (2024). The quantum frontier of software engineering: A systematic mapping study. *Information and Software Technology*, page 107525.

Exman, I., Pérez-Castillo, R., Piattini, M., and Felderer, M. (2024). *Quantum Software: Aspects of Theory and System Design*. Springer Nature.

Ezratty, O. (2024). Understanding quantum technologies 2024.

Fadhlillah, H. S. and Rabiser, R. (2024). Towards a product configuration representation for the universal variability language. In *Proceedings of the 28th ACM International Systems and Software Product Line Conference*, pages 50–54.

Faro, I., Sitdikov, I., Valiñas, D. G., Fernandez, F. J. M., Codella, C., and Glick, J. (2023). Middleware for quantum: An orchestration of hybrid quantum-classical systems. In *2023 IEEE International Conference on Quantum Software (QSW)*, pages 1–8. IEEE.

Ferrara, R., Bassoli, R., Deppe, C., Fitzek, F. H., and Boche, H. (2021). The computational and latency advantage of quantum communication networks. *IEEE Communications Magazine*, 59(6):132–137.

Galindo, J. A., Benavides, D., Trinidad, P., Gutiérrez-Fernández, A.-M., and Ruiz-Cortés, A. (2019). Automated analysis of feature models: Quo vadis? *Computing*, 101:387–433.

Garcia-Alonso, J., Rojo, J., Valencia, D., Moguel, E., Berrocal, J., and Murillo, J. M. (2021). Quantum software as a service through a quantum api gateway. *IEEE Internet Computing*, 26(1):34–41.

Grange, C., Poss, M., and Bourreau, E. (2023). An introduction to variational quantum algorithms for combinatorial optimization problems. *4OR*, 21(3):363–403.

Heim, B., Soeken, M., Marshall, S., Granade, C., Roetteler, M., Geller, A., Troyer, M., and Svore, K. (2020). Quantum programming languages. *Nature Reviews Physics*, 2(12):709–722.

Hevia, J. L., Peterssen, G., and Piattini, M. (2024). qsoa®: Dynamic integration for hybrid quantum/classical software systems. *Journal of Systems and Software*, 214:112061.

Kahn, J. (2020). D-wave unveils its most powerful quantum computer to date. `https://fortune.com/2020/09/29/d-wave-5000-qubit-quantum-computer/`. Accessed on January 25, 2025.

K.S., J. and Gowda, S. (2023). Quantum network for quantum communication. In *2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–5.

LaRose, R. (2019). Overview and Comparison of Gate Level Quantum Software Platforms. *Quantum*, 3:130.

McGeoch, C. C. (2020). Theory versus practice in annealing-based quantum computing. *Theoretical Computer Science*, 816:169–183.

Nadim, M., Hassan, M., Mandal, A. K., and Roy, C. K. (2024). Quantum vs. classical machine learning algorithms for software defect prediction: Challenges and opportunities. `https://arxiv.org/abs/2412.07698`. ArXiv preprint.

Nielsen, M. A. and Chuang, I. L. (2001). *Quantum computation and quantum information*, volume 2. Cambridge university press Cambridge.

Pendenti, H., Yepez, L. P., Castillo, R. P., Alegría, J. A. H., Antonelli, L., and Fernandez, A. (2024). ¿qué tiene de especial la ingeniería de requisitos para los sistemas cuántico-clásicos? *Memoria Investigaciones en Ingeniería*, 1(27):257–265.

Piattini, M., Peterssen, G., Pérez-Castillo, R., Hevia, J. L., Serrano, M. A., Hernández, G., De Guzmán, I. G. R., Paradela, C. A., Polo, M., Murina, E., et al. (2020). The talavera manifesto for quantum software engineering and programming. In *QANSWER*, pages 1–5.

Pérez-Castillo, R., Serrano, M. A., and Piattini, M. (2021). Software modernization to embrace quantum technology. *Advances in Engineering Software*, 151:102933.

Qi, H., Xiao, S., Liu, Z., Gong, C., and Gani, A. (2024). Variational quantum algorithms: fundamental concepts, applications and challenges. *Quantum Information Processing*, 23(6):224.

Rabiser, R. (2024). Industry adoption of uvl: What we will need. In *Proceedings of the 28th ACM International Systems and Software Product Line Conference*, pages 46–49.

Rojo, J., Valencia, D., Berrocal, J., Moguel, E., Garcia-Alonso, J., and Rodriguez, J. M. M. (2021). Trials and tribulations of developing hybrid quantum-classical microservices systems. *arXiv preprint arXiv:2105.04421*.

Safdar, S. A., Yue, T., Ali, S., and Lu, H. (2016). Evaluating variability modeling techniques for supporting cyber-physical system product line engineering. In *System Analysis and Modeling. Technology-Specific Aspects of Models: 9th International Conference, SAM 2016, Saint-Melo, France, October 3-4, 2016*, pages 1–19. Springer.

Salam, M. and Ilyas, M. (2024). Quantum computing challenges and solutions in software industry—a multivocal literature review. *IET Quantum Communication*.

Salm, M., Barzen, J., Leymann, F., Weder, B., and Wild, K. (2021). Automating the Comparison of Quantum Compilers for Quantum Circuits. In *Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2021)*, pages 64–80. Springer International Publishing.

Schönberger, M., Franz, M., Scherzinger, S., and Mauerer, W. (2022). Peel | pile? cross-framework portability of quantum software.

Sekavčnik, S. and Nötzel, J. (2022). Effects of quantum communication in large-scale networks at minimum latency. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 802–807.

Sepúlveda, S., Pérez-Castillo, R., and Piattini, M. (2025). A software product line approach for developing hybrid software systems. *Information and Software Technology*, 178:107625.

Sepúlveda, S., Piattini, M., and Pérez Del Castillo, R. (2024). Developing hybrid quantum-classical software: a software product line approach. In *Proceedings of the 5th ACM/IEEE International Workshop on Quantum Software Engineering (Q-SE)*, page 37–40, New York, USA. ACM.

Sepúlveda, S., Perez-Castillo, R., and Piattini, M. (2025). Feature model hqc-spl cibse2025. `https://doi.org/10.5281/zenodo.15013631`. DOI: 10.5281/zenodo.15013631.

Serrano, M. A., Pérez-Castillo, R., and Piattini, M. (2022). *Quantum software engineering*. Springer.

Stirbu, V. and Mikkonen, T. (2023). Software architecture challenges in integrating hybrid classical-quantum systems. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 2, pages 203–204. IEEE.

Zhao, J. (2021). Quantum software engineering: Landscapes and horizons. `https://arxiv.org/abs/2007.07047`. ArXiv preprint.