Una adaptación del UML Testing Profile para el Testing de Software Cuántico

An Adaptation of the UML Testing Profile for Quantum Software Testing

Uma Adaptação do Perfil de Teste UML para Teste de Software Quântico

Ignacio García Rodríguez de Guzmán [1,(*)], Moisés Rodríguez Monje [2],
Mario Piattini Velthuis [3], María Teresa Merchan Quintana [4]

Recibido: 10/10/2024 Aceptado: 10/10/2024

Resumen. - Este estudio explora cómo la Ingeniería del Software Clásico, especialmente la ingeniería dirigida por modelos en testing, puede contribuir a la mejora de la Ingeniería del Software Cuántico (Quantum Software Engineering, QSE). Adoptando principios de "agnosticismo", se busca desarrollar procesos de ingeniería del software que sean independientes de cualquier lenguaje o plataforma específicos para software cuántico. Actualmente, el testing de software cuántico se considera un área emergente con múltiples propuestas focalizadas en lenguajes o plataformas particulares. Este trabajo propone una extensión del UML Testing Profile 2.0 para modelar pruebas de software cuántico, permitiendo (i) la exploración de nuevos conceptos necesarios para especificar estas pruebas, (ii) el modelado de diferentes tipos de pruebas durante el diseño del software cuántico, y (iii) la especificación de pruebas independiente del lenguaje o plataforma de ejecución. Esta aproximación busca establecer una base más flexible y extensible para el testing en QSE.

Palabras clave: Ingeniería de software cuántica; Pruebas de software cuánticas; Perfil de pruebas UML; Pruebas de software.

(*) Autor de correspondencia.

[1] Catedrático de Universidad, Universidad de Castilla-La Mancha. Ignacio.GRodriguez@uclm.es,
ORCID iD: https://orcid.org/0000-0002-0038-0942

[2] Profesor Titular de Universidad, Universidad de Castilla-La Mancha. Moises.Rodriguez@uclm.es,
ORCID iD: https://orcid.org/0000-0003-2155-7409

[3] Profesor Titular de Universidad, Universidad de Castilla-La Mancha. Mario.Piattini@uclm.es,
ORCID iD: https://orcid.org/0000-0002-7212-8279

[4] Estudiante de Máster, Universidad de Castilla-La Mancha. MTeresa.Merchan@alu.uclm.es,
ORCID iD: https://orcid.org/0009-0005-7162-8270

Memoria Investigaciones en Ingeniería, núm. 27 (2024). pp. 266-284
https://doi.org/10.36561/ING.27.18
ISSN 2301-1092 • ISSN (en línea) 2301-1106 – Universidad de Montevideo, Uruguay

Summary. - *This study explores how Classical Software Engineering, especially model-driven engineering in testing, can contribute to the improvement of Quantum Software Engineering (QSE). Adopting principles of "agnosticism", it seeks to develop software engineering processes that are independent of any specific language or platform for quantum software. Currently, quantum software testing is considered an emerging area with multiple proposals focused on particular languages or platforms. This work proposes an extension of the UML Testing Profile 2.0 to model quantum software testing, allowing (i) the exploration of new concepts needed to specify these tests, (ii) the modelling of different types of tests during quantum software design, and (iii) the specification of tests independent of the execution language or platform. This approach aims to establish a more flexible and extensible basis for testing in QSE.*

Keywords: *Quantum Software Engineering; Quantum Software Testing; UML Testing Profile; Software Testing*

Resumo. ***-*** *Este estudo explora como a Engenharia de Software Clássica, especialmente a engenharia orientada por modelos em testes, pode contribuir para a melhoria da Engenharia de Software Quântica (QSE). Adoptando princípios de "agnosticismo", procura desenvolver processos de engenharia de software que sejam independentes de qualquer linguagem ou plataforma específica para software quântico. Atualmente, o teste de software quântico é considerado uma área emergente com múltiplas propostas centradas em linguagens ou plataformas específicas. Este trabalho propõe uma extensão do Perfil de Testes UML 2.0 para modelar testes de software quântico, permitindo (i) a exploração de novos conceitos necessários para a especificação destes testes, (ii) a modelação de diferentes tipos de testes durante a conceção de software quântico, e (iii) a especificação de testes independentes da linguagem ou plataforma de execução. Esta abordagem tem como objetivo estabelecer uma base mais flexível e extensível para os testes em QSE.*

Palavras-chave: *Engenharia de Software Quantum; Teste de Software Quantum; Perfil de Teste UML; Teste de Software.*

1. Introduction. - Aunque la computación cuántica ha irrumpido con fuerza en el panorama actual con augurios de resolver problemas que, hasta el momento, se plantean como retos fuera del alcance de la tecnología "clásica". Sin embargo, no debemos olvidar que nos encontramos en los albores de un paradigma tecnológico que de la misma forma que la computación clásica, comenzó con altos niveles de inmadurez.

En este momento, los principios de la Ingeniería del Software Cuántico (QSE) [12] establecen un *roadmap* a seguir mediante el cual, llegar a generar conocimiento, técnicas y herramientas mediante las cuales hacer sostenible el desarrollo de software cuántico, evitando así caer en potenciales crisis (del software cuántico) resultantes de la diferencia entre la expectativa generada por la computación cuántica y la capacidad real de la industria para suministrar soluciones.

Uno de los retos que están surgiendo irremediablemente, es la aparición de múltiples lenguajes y plataformas de ejecución de software cuántico (simuladores y computadores cuánticos de distintos proveedores) con sus propias características y particularidades. Las plataformas de ejecución plantean serias diferencias tecnológicas entre sí como, por ejemplo, algo tan básico como la topología de sus QPU (*Quantum Processor Unit*) o los juegos de instrucciones básicos (puertas cuánticas disponibles para ejecutar el software cuántico).

La heterogeneidad en los sistemas software no es un reto desconocido para la industria y la academia, que ya ha propuesto soluciones eficaces a la hora de tratar con esta situación. La ingeniería dirigida por modelos [6] establece una pila de niveles que van, desde vistas de muy alto nivel (totalmente agnósticas a la tecnología subyacente), hasta modelos muy cercanos a la plataforma para la cual se generará el código ejecutable. Este paradigma, resulta de especial interés para la QSE donde (i) el agnosticismo es uno de los principios básicos aplicables a todas sus áreas, y (ii) donde la heterogeneidad de sus actuales propuestas motiva la necesidad de mecanismos que permitan trabajar con software cuántico con independencia de los servicios de ejecución donde el software cuántico vaya a ser ejecutado (ya sea incluso un simulador cuántico, los cuales suelen ser más flexibles y desde donde es a veces difícil exportar el código directamente a un computador cuántico).

Uno de los aspectos de la QSE qué tiene más impacto en la calidad del software cuántico (y que convendría abordar por su relevancia [11, 13]), donde el modelado podría resultar beneficioso, es el testing. En la actualidad, y aunque comienzan a surgir propuestas de testing orientadas a la QSE[14], el estado del testing cuántico es precario [3]. Por lo que resulta necesario comenzar a potenciar este proceso desde la base, considerando tanto la definición de conceptos básicos como estandarización de los principios. Así, el testing cuántico sería independiente de las tecnologías y las plataformas cuánticas.

El testing del software, como proceso clásico de la Ingeniería del Software, se organiza en base a un conjunto de conceptos, actividades y procesos comunes. Todos estos elementos vienen descritos en el estándar ISO 29119 [8], que constituyen (en sus partes I y II) unas guías de referencia para la identificación de los (i) conceptos relativos al proceso de testing y (ii) sus procesos principales.

A partir de la norma ISO 29119 podemos encontrar representaciones basadas en el paradigma de ingeniería dirigida de modelos (*Model Driven Engineering*, MDE), mediante las cuales, poder diseñar y generar los artefactos de prueba dentro de los ciclos de desarrollo del software, aportando (i) altos niveles de automatización e (ii) independencia del proceso de prueba con respecto a la tecnología bajo la que se desarrollan los SUT (*System Under Test*). Esta característica de la MDE, en el contexto del testing del software, aporta ese agnosticismo que se promueve para QSE.

En al ámbito del testing del software y la MDE, existe un metamodelo/perfil denominado UML Testing Profile 2.0 (UMLTP) [9], que tiene como objetivo establecer el conjunto de conceptos relativos al testing del software (con un gran paralelismo a los identificados por la ISO 29119), organizados en forma de metamodelo, y basado en el metamodelo UML 2.0 para la especificación de sistemas software.

Llegados a este punto, en este trabajo se propone partir de este artefacto, de aplicabilidad ya probada en el mundo de la industria, para su análisis y extensión de forma que, al igual que ocurre con el software clásico, el software cuántico

pueda beneficiarse del diseño, modelado y generación de los artefactos involucrados en el proceso de testing. Los beneficios de esta extensión garantizarían: (i) el agnosticismo del proceso de testing (en el marco de la QSE); (ii) una mejora en el estado del arte con respecto a herramientas y metodologías para el desarrollo de software cuántico y, (iii) una oportunidad para avanzar hacia el testing de sistemas híbridos (desarrollados por software cuántico y clásico), ya que esta extensión cubriría los artefactos de testing de ambos paradigmas).

Así, en este artículo se presenta:

- Estudio sobre los conceptos relativos el proceso de testing adicionales a los existentes en la ISO/IEC 29119. Dado que UMLTP está basado en este estándar para la definición de su conjunto de estereotipos, cualquier extensión del UMLTP debería por lo tanto basarse en una versión de la ISO/IEC 29119 que también considerara la computación cuántica. Al no existir dicha extensión o versión de la norma, se propone por lo tanto un conjunto de términos novedosos sobre los cuales, plantear la posterior extensión del UMLTP.
- Propuesta de un conjunto nuevo de estereotipos para el UMLTP basado en los nuevos conceptos definidos para la norma ISO/IEC 29119. Dicha propuesta, se realiza en base a las distintas vistas en las que se organiza el UMLTP.

Este artículo se organiza de la siguiente forma: la sección II presenta un breve estado del arte donde se dan algunas nociones sobre el UML Testing Profile, y se revisan algunas de las propuestas de taxonomía relacionadas con el testing del software clásico; la sección III realiza un breve análisis de la norma ISO 29119; la sección IV presenta la propuesta de extensión de UMLTP para incluir conceptos relativos al software cuántico necesarios para modelar pruebas en este paradigma; finalmente, la sección V presenta algunas conclusiones.

2. Estado del Arte. - Esta sección presenta, algunos aspectos relevantes al trabajo como son: (i) la descripción del UML Testing Profile 2.0, que se toma como perfil candidato a extender para poder modelar artefactos de prueba en el software cuántico; y (ii) distintas taxonomías del testing clásico, donde se proponen distintas clasificaciones para los elementos, técnicas y tecnologías relacionadas con el testing del software. El estado del arte, por lo tanto, se enfoca en revisar aquellos aspectos relativos a la clasificación de conceptos del testing y su extensión al paradigma del software cuántico.

2.1. UML Testing Profile.. - El "UML Testing Profile 2.0" (UMLTP) es una especificación desarrollada por el OMG que define cómo modelar pruebas de software dentro del entorno de la Lenguaje Unificado de Modelado (UML). El perfil se diseñó para integrar y adaptar los conceptos tradicionales de pruebas de software al modelado UML, con el objetivo de facilitar un enfoque más coherente y estandarizado para el diseño y ejecución de pruebas.

UTP 2.0 extiende UML para abordar necesidades específicas de las pruebas, proporcionando elementos de modelado que representan casos de prueba, suites de pruebas, estrategias de pruebas, y otros artefactos relacionados con las pruebas. Además, permite la descripción de la estructura de las pruebas, su comportamiento, y los resultados esperados de manera que puedan ser gestionados y comprendidos dentro del ciclo de vida del desarrollo de software.

Uno de los principales componentes de UTP 2.0 es su capacidad para modelar tanto pruebas estructurales como de comportamiento. Esto incluye la capacidad de definir pruebas a nivel de componente y a nivel de sistema, lo que permite a los ingenieros de prueba especificar cómo deben interactuar los componentes entre sí y cómo deben comportarse en el contexto de sistemas más amplios.

El perfil también se integra con otros perfiles de UML para proporcionar una vista más completa del desarrollo y pruebas de sistemas. Esto facilita una mejor colaboración entre los equipos de desarrollo y pruebas, asegurando que los modelos de pruebas estén alineados con los modelos de desarrollo y los requisitos del sistema.

UTP 2.0 es una herramienta valiosa para los desarrolladores y testers que utilizan UML para diseñar y documentar sistemas de software, ya que proporciona una metodología robusta para integrar las pruebas en el proceso de modelado, asegurando que las pruebas se consideren como parte integral del proceso de desarrollo de software desde las primeras etapas.

2.2. Taxonomías de Testing Clásico. - El desarrollo de una extensión para el modelado de artefactos de testing del software cuántico implica partir de una representación adecuada, donde los conceptos relevantes deben estar presentes. En la literatura se pueden encontrar varias clasificaciones y/o taxonomías que recogen dichos conceptos así como sus relaciones.

En [5], los autores proponen una taxonomía donde se identifican los elementos presentes en el testing desde 6 perspectivas: Tester, contexto, actividad, método, artefactos y entorno (ver Figura I). Aunque dicha taxonomía cubre muchos de los conceptos y técnicas presentes en el proceso de testing, no aborda los procesos que sí se consideran en la ISO 29119, así como otros conceptos relevantes.

Por otro lado, en [15] se presenta una taxonomía de herramientas para automatizar el testing del software. Dicho trabajo se centra en las herramientas de testing, donde a priori las clasifica en tres grandes categorías categorías: testing funcional, testing de gestión y loading testing (para cada una de las cuales distingue entre herramientas *open source* o comerciales). Posteriormente y en un intento por ayudar a clasificar las herramientas, este trabajo propone un conjunto de criterios para evaluar las herramientas y poder así clasificarlas: soportado por navegador web, requiere licencia, dispone de sitio web con información, precio, herramientas soportadas, lenguajes de programación que soporta. Aunque a priori, esta taxonomía permite clasificar herramientas, no sirve en tanto en cuanto no contribuye a estandarizar el proceso de testing. Sin embargo, en [2] se presenta una clasificación de los distintos tipos de testing, lo cual desde el punto de vista de la estandarización, sí que puede ser más representativo. En este trabajo, se identifican 7 grandes tipos (ver Figura II): el testing de modelos, el testing del software, el testing de centros de datos, el testing de entornos o herramientas, el testing del hardware, y el testing de sistemas.

En [16], se focaliza exclusivamente en el testing de proyectos software, para lo que se proponen 8 categorías: proyectos de escritorio, web, dispositivos móviles, servicios, procesos, control del tiempo, migración de sistemas, almacenamiento y protocolos. Aunque el trabajo despliega posteriormente esta primera clasificación en categorías concretas (ver ejemplo de ello en Figura III), no se tiene en cuenta otros conceptos más genéricos y básicos del proceso del testing, lo cual puede influir en gran medida en cómo se caracterizan dichos tipos proyecto.

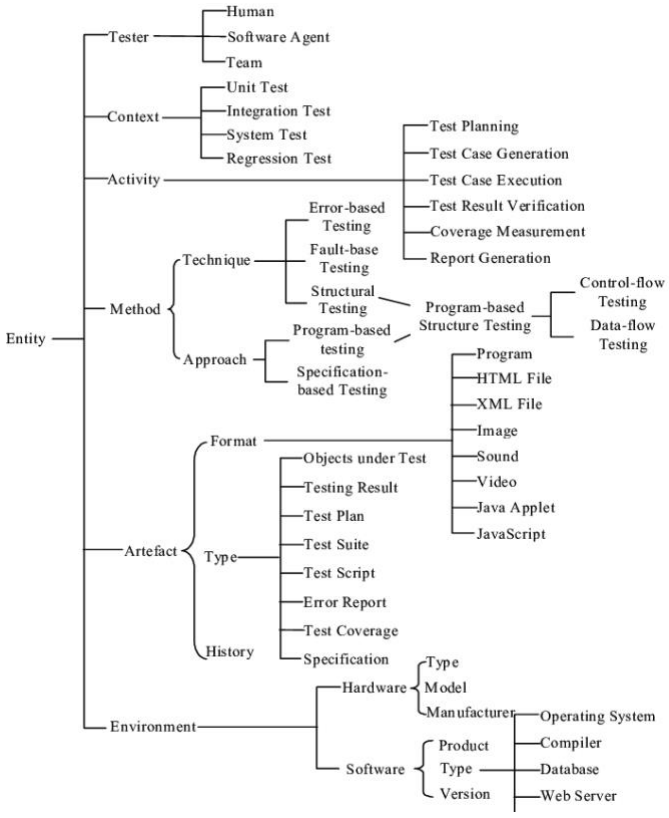


Figura I. Taxonomía del cencepto del testing del software [5]

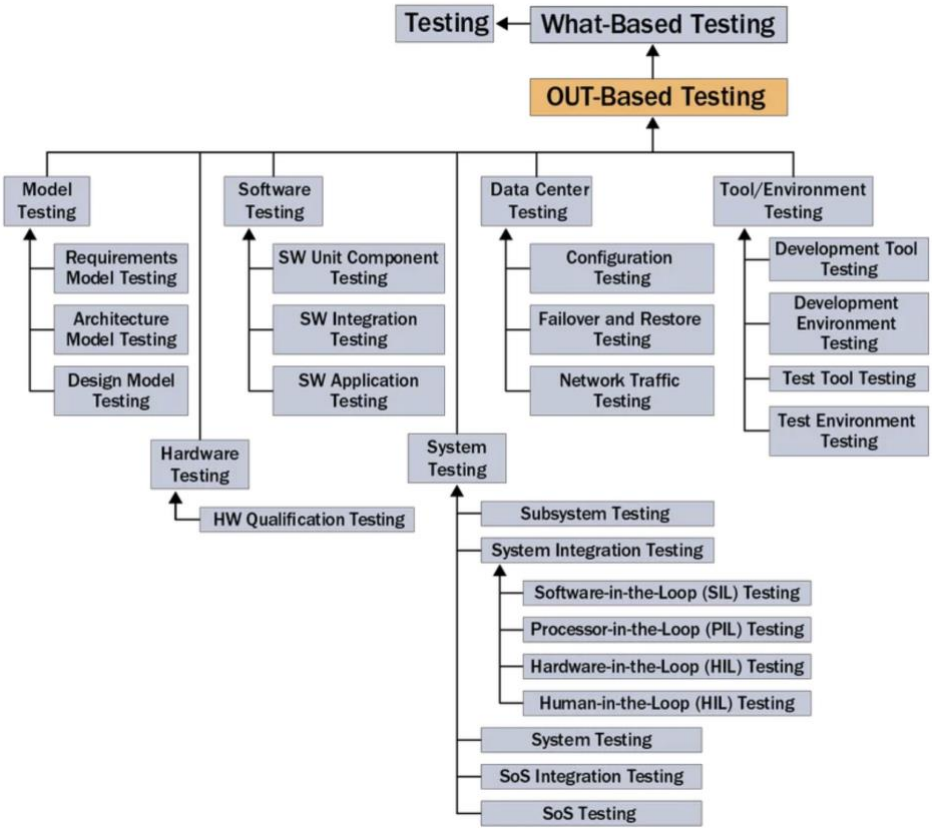


Figura II. Taxonomía de los tipos de testing [2]

Categoría	Sub-categoría
Web	Páginas o sitios web
	Webs tradicionales
	Rich Internet Applications (RIA)
	Web 2.0

Figura III. Taxonomía de los tipos de testing [16]

Tal y como se ha podido observar, las taxonomías o clasificaciones existentes se centran en aspectos concretos del testing del software que podrían incluso ser transversales al paradigma computacional en el que se desarrolle en software. Por este motivo, consideramos que el estándar ISO 29119 resulta ser la fuente más fiable para comenzar a replantear el conjunto de elementos del testing que, de cara al testing de software cuántico, deben ser mantenidos, redefinidos, o agregados para poder modelar los artefactos de prueba para este nuevo paradigma.

3. Análisis del estándar ISO 29119. - El metamodelo UML Testing Profile 2.0 representa, mediante una notación de modelo (y con propósito de modelado) los distintos conceptos relacionados con el testing de software. Dichos conceptos, están estandarizados y así se recogen la mayoría en la norma ISO/IEC 29119-1 [7], aunque sería más correcto decir que fruto de la estandarización del proceso de testing del software y de sus términos relacionados, se ha desarrollado el UMLTP, que permite llevar una gestión de los artefactos de prueba de un sistema bajo prueba a nivel de modelo.

Por este motivo, como primer paso para la extensión del UMLTP se ha llevado a cabo un estudio de la parte 1 del estándar ISO 29119, donde se establecen los conceptos entorno a los cuales se describen los distintos procesos del testing de software. De estos conceptos, posteriormente, se han propuesto especializaciones en el UMLTP, en base a la necesidad de incluirlos en los modelos según su relevancia o existencia en la literatura.

3.1. Nuevos conceptos en base al estándar. - A continuación, en la TABLE I se describen los nuevos conceptos propuestos para el estándar ISO/IEC 29119, indicando en su caso el concepto original del cual se derivan.

Concepto Original	Concepto Derivado	Definición
Actual results (3.6)	Quantum actual results	Conjunto de comportamientos o condiciones de un *Quantum Test Item*, o conjunto de condiciones de datos asociados o el entorno de prueba, observado como resultado de la ejecución de los tests, y que se corresponderían con los datos de telemetría del simulador o computador cuántico.
Simulator (3.154)	Quantum Simulator	Entorno de prueba, sobre el cual se debe tener en cuenta que no funciona como un sistema físico puro, y del cual es necesario conocer cualquier parámetro sobre su funcionamiento con respecto a la desviación que presenta con respecto a un computador cuántico.
-	Shot	Ejecución única de un sistema bajo prueba en el contexto de un caso de prueba.
-	Statistic testing	Técnica de prueba para sistemas cuánticos, mediante la cual se realiza un análisis del resultado del caso de prueba cuántico en base a un número de *shots*, para verificar el comportamiento esperado.
Test case (3.172)	Quantum Test Case	Caso de prueba específico para sistemas cuánticos, donde entre otros parámetros, hay que especificar el número de *shots* requeridos.
Test case (3.172)	Deterministic Test Case	Caso concreto de prueba para circuitos cuánticos deterministas, donde se espera un resultado concreto, un valor único derivado de un circuito sin comportamiento estocástico.
Test case (3.172)	Non-Deterministic Test Case	Caso concreto de prueba para circuitos cuánticos no deterministas, donde se espera un resultado expresado como una distribución de probabilidades. Un circuito estocástico, según su definición, podría incluso tener un comportamiento determinista.
Test coverage (3.177)	Quantum Test Coverage	Grado, expresado como un porcentaje, en el cual se han ejecutado los distintos *Quantum Test Coverage Item* del circuito cuántico por el efecto de un *Quantum Test Case*.
Test coverage item (3.178)	Quantum Test Coverage Item	Cualquiera de los artefactos que pueden emplearse para la definición de un circuito (puerta, oráculos, cúbits, etc.)
Test data (3.179)	Quantum Test Data	Definición de un conjunto de estados de los qubits requeridos para la ejecución del circuito cuántico bajo prueba.
Test environment (3.184)	Test quantum environment	Entorno o servicio de ejecución de software cuántico necesario para ejecutar un test cuántico. Incluirá un Quantum computer o un Quantum Simulator. Incluirá un entorno de ejecución clásico, desde el cual se conducirá la prueba del software cuántico.

Test environment item (3.185)	Quantum test environment item	Parte de un *test quantum environment* que tiene como fin exclusivo la ejecución de software cuántico. Puede ser un *Quantum Simulator*, ejecutado sobre un entorno clásico o un *Quantum computer*.
Test item (3.199)	Quantum test item	Producto de carácter cuántico que será testeado (circuito cuántico, código cuántico, software híbrido, etc.).
Test model (3.203)	Quantum test model	Representación de un *quantum test item*, que permite focalizar el proceso de testing en un circuito completo o una de sus partes integrantes.
Test Oracle problema (3.208)	Quantum test Oracle problema strategy	Mecanismo para determinar en qué medida un *Quantum Test Case* ha pasado o fallado en el contexto de sus condiciones de cumplimiento. Para condiciones deterministas las estrategias son (i) la prueba del SWAP y la prueba del NOT.
Test result (3.218)	Quantum test result	En el caso de software cuántico hay que comparar el resultado obtenido por el *quantum test case* con el resultado esperado. Esta comprobación variará según la naturaleza del circuito.

Tabla I. Conceptos de testing cuántico derivados de la ISO/IEC 29119

4. Definición de nuevas etiquetas para el UML Testing Profile 2.0. –

4.1. Estrategia de extensión. - La estrategia a seguir comienza con el estudio de la especificación de UML Testing Profile (UMLTP), extrayendo cada una de las vistas de las cuales se compone, y que serán analizadas para identificar (i) los elementos que deben ser extendidos o (ii) añadidos.

La eliminación de estereotipos existentes del UMLTP no se contempla, ya que, en un futuro, el testing de sistemas híbridos (software clásico y cuántico) requerirá de la cobertura necesaria para representar artefactos de prueba de ambos paradigmas.

Una vez que se seleccionen las vistas relevantes sobre las que se van a realizar las extensiones, se presentarán qué partes de las mismas son susceptibles de extender, generando una nueva versión con aquellas clases adicionales.
Las vistas seleccionadas para su extensión y adaptación para testing cuántico son:

- *Test Context Overview*: Esta vista es una representación que resume los aspectos esenciales del contexto de las pruebas de software. Incluye elementos como actores de prueba, casos de prueba, objetivos de prueba, artefactos de prueba, niveles de prueba y estrategias de prueba. Además, proporciona una visión general de cómo se estructuran y relacionan estos elementos en el proceso de prueba, lo que facilita la comprensión y comunicación de los diferentes aspectos involucrados en el diseño y ejecución de pruebas utilizando el UMLTP.
- *Test Architecture Overview*: La vista Test Architecture Overview, proporciona una descripción de la arquitectura de pruebas de software utilizando el perfil de prueba de UMLTP. Incluye elementos como componentes de prueba, interacciones entre los componentes, interfaces de prueba, relaciones de dependencia y estructuras de datos de prueba. Esta vista brinda una visión general de cómo se organizan y relacionan estos elementos en la arquitectura de pruebas, lo que facilita la comprensión y comunicación de la estructura y el diseño de las pruebas en el modelo UMLTP.
- *Test Case Overview*: La vista Test Case Overview proporciona una descripción de los casos de prueba en el proceso de prueba de software utilizando el perfil de prueba de UML. Incluye elementos como nombres de casos de prueba, descripciones, precondiciones, acciones, resultados esperados y datos de prueba. Ofrece una visión general de cómo se estructuran y organizan los casos de prueba, facilitando la comprensión y comunicación de los escenarios de prueba y sus objetivos dentro del modelo UMLTP.

- *Test Log Overview*: Esta vista consiste en una representación del registro de pruebas en el proceso de prueba de software utilizando el perfil de prueba de UML. En esta vista se incluyen elementos como registros de prueba, resultados obtenidos, fechas y horarios de ejecución, pasos de prueba realizados, errores detectados y acciones de seguimiento tomadas. Proporciona una visión general de cómo se registra y documenta el progreso y los resultados de las pruebas, lo que facilita la comprensión y comunicación de los resultados obtenidos, los problemas identificados y las acciones correctivas necesarias en el contexto del modelo UMLTP.
- *Arbitration and Veredict Overview*: Esta vista proporciona una descripción del proceso de arbitraje y veredicto en el contexto de las pruebas de software utilizando el perfil de pruebas de UML. Incluye elementos como la resolución de conflictos, la toma de decisiones y la evaluación de los resultados de las pruebas para determinar el veredicto final. Esta vista ofrece una visión general de cómo se maneja el proceso de toma de decisiones y se emite un veredicto basado en los resultados de las pruebas en el marco del UMLTP.

4.2. Extensión de la vista "*Test Context OverView*". - En esta vista, se representan los casos de prueba, los elementos de prueba y los actores de prueba junto con sus correspondientes relaciones. También se pueden mostrar las dependencias entre los casos de prueba y los elementos del sistema que están siendo probados. Además, se pueden incluir restricciones y reglas de negocio que se aplican a los casos de prueba.

La vista "Test Context Overview" es útil para comprender rápidamente como se estructuran las pruebas en un sistema y cómo se relacionan con los otros componentes. Proporciona una representación visual clara y concisa del contexto de prueba, lo que facilita la comunicación entre los miembros del equipo de desarrollo y el equipo de pruebas.

A continuación, en la Figura IV, que se puede observar a alto nivel, las distintas partes en las que se divide la vista "Test Context Overview".

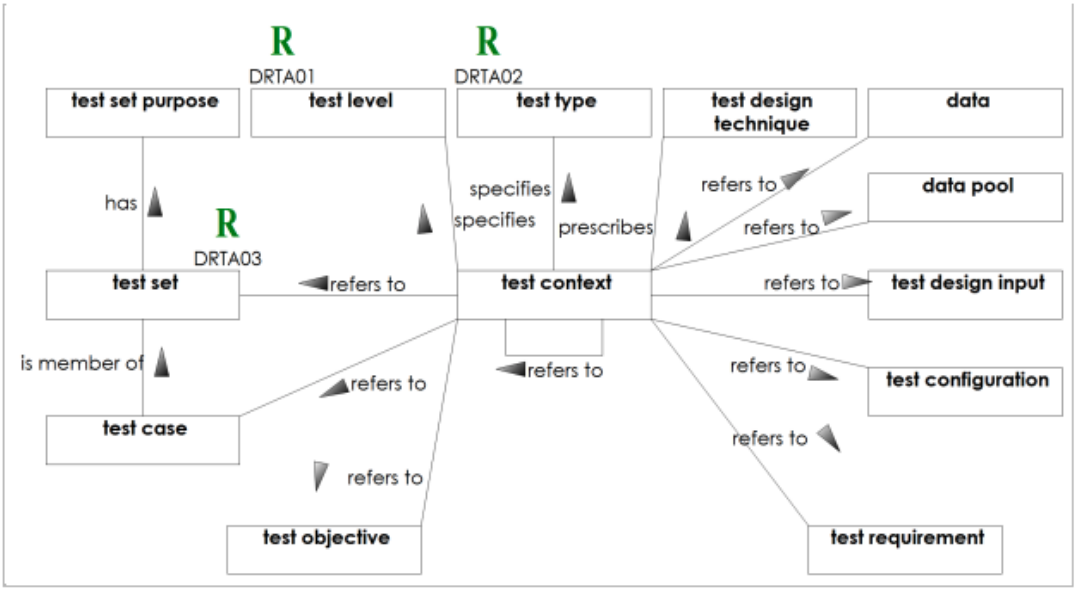


Figura IV. Representación de alto nivel de la vista "Test Context Overview".

La vista presentada en la Figura IV muestra una vista de alto nivel. Los estereotipos (y clases) mediante las cuales se ha realizado la extensión de los estereotipos aplicables de la vista "*Test Context Overview*", generando una nueva versión de la misma dando cobertura a los nuevos conceptos propios del testing de software cuántico basado en circuitos quedan reflejados en la en la Figura V.

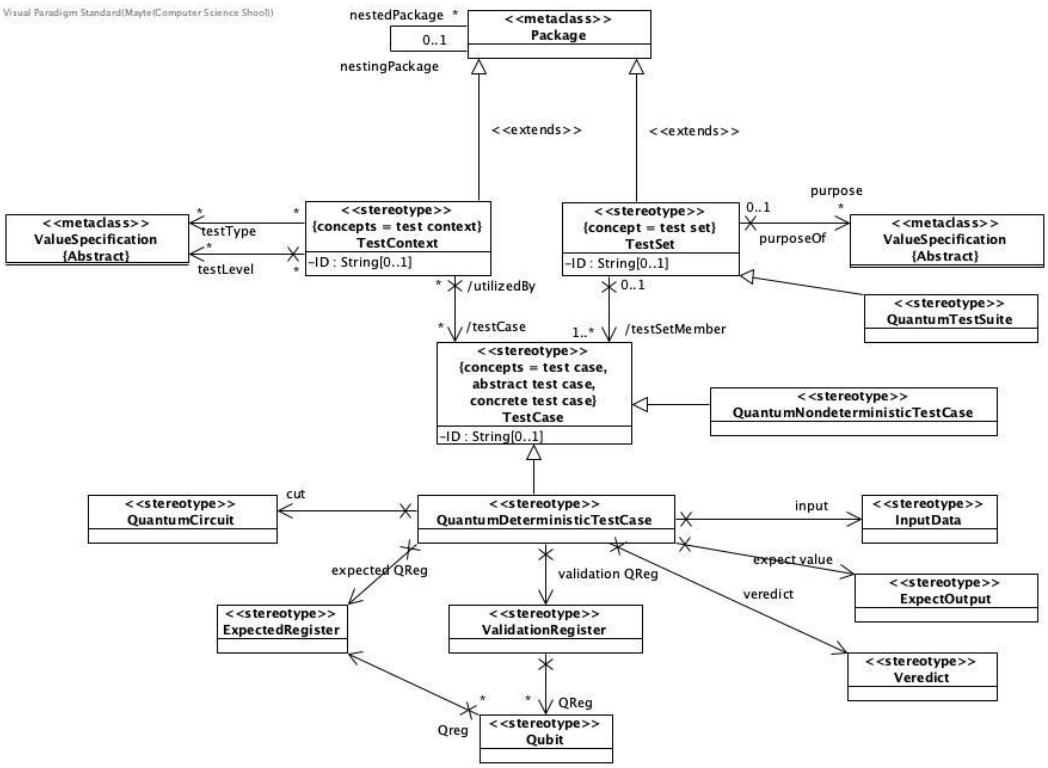


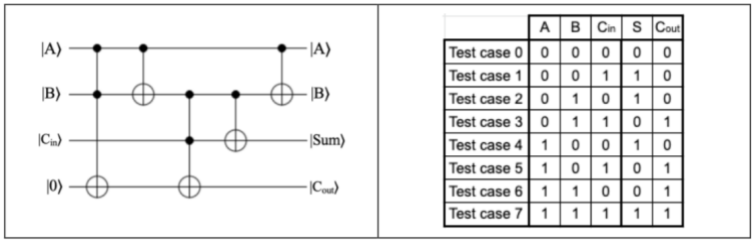
Figura V. Extensión de la vista "Test Context Overview"

El estereotipo *QuantumTestSuite* tiene como objetivo representar al concepto de Test Suite en el ámbito del testing cuántico. Es decir, este Estereotipo representa al contenedor del conjunto de casos de pruebas para el software cuántico. Su representación, a diferencia de un caso de prueba clásico puede variar en tanto en cuanto consideremos la representación del caso de prueba cuántico.

El concepto de caso de prueba cuántico es novedoso en el contexto del software cuántico, ya que la naturaleza disruptiva del software cuántico hace que se requieran artefactos distintos para su prueba. Incluso la formulación del mismo podría resultar muy distinta del propio concepto de prueba clásico. A esto, debemos sumar la propia naturaleza de los circuitos cuánticos, que podrían clasificarse en dos categorías según el comportamiento de los qubits que integran los circuitos: circuitos cuánticos deterministas y no deterministas. En esta sección, consideramos la posibilidad de tener que testear circuitos clásicos deterministas, es decir, aquellos en los que los qubits del mismo no se encuentran en superposición, y cuyo resultado no será una distribución de probabilidades.

Por ello, se añade el nuevo Estereotipo *QuantumDeterministicTestCase*, que representa al concepto de "Caso de prueba" para circuitos cuánticos deterministas.

En Figura VI (izda.) se puede observar un ejemplo de software cuántico, modelado en este caso como un circuito determinista. Por otro lado, la Figura VI (dcha.) representaría los casos de prueba que conforman una QuantumTestSuite, compuesta por un conjunto de casos de QuantumDeterministicTestCase para circuitos cuánticos de este tipo.



	A	B	C_{in}	S	C_{out}
Test case 0	0	0	0	0	0
Test case 1	0	0	1	1	0
Test case 2	0	1	0	1	0
Test case 3	0	1	1	0	1
Test case 4	1	0	0	1	0
Test case 5	1	0	1	0	1
Test case 6	1	1	0	0	1
Test case 7	1	1	1	1	1

Figura VI. Sumador cuántico con acarreo(izquierda) y test suite del sumador(derecha)

Así mismo y de este modo, la Figura VII presenta los estereotipos que modelan el caso de prueba determinístico, ya que para este caso concreto, sí existen propuestas relativas a este tipo de casos de prueba cuánticos [1]. Los estereotipos que se proponen para describir este tipo de caso de prueba son los siguientes:

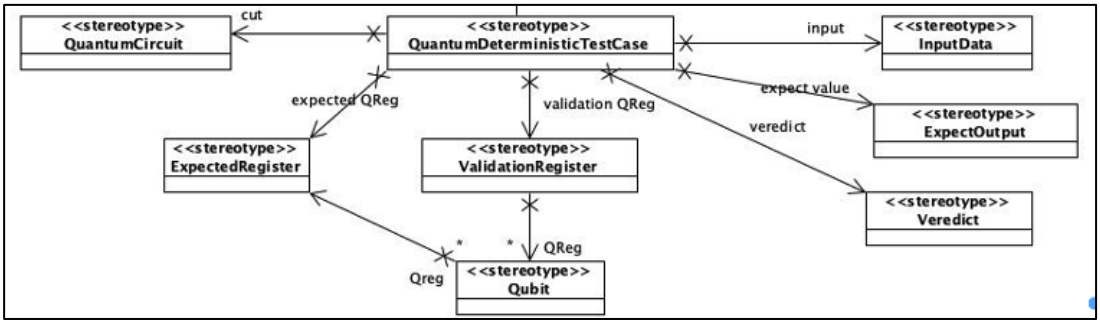


Figura VII. Estereotipos para describir un QuantumDeterministicTestCase.

- Estereotipo *ExpectedRegister*: El concepto de *ExpectedRegister* en un circuito cuántico se refiere al estado o resultado esperado del registro cuántico después de que se hayan aplicado un conjunto de operaciones cuánticas. En computación cuántica, un registro cuántico en una colección de cúbit. Por ello, se añade la clase *ExpectedRegister* que tiene como objetivo almacenar los valores de los cúbit tras la ejecución del circuito cuántico.
- Estereotipo *InputData*: La clase *InputData* se añade con el fin proporcionar los datos de entrada que van a ser utilizados como información inicial en un circuito cuántico, siendo esenciales para obtener resultados correctos o deseados en el mismo. Los valores de entrada representan los estados cuánticos de los qubits de entrada al circuito.
- Estereotipo *ExpectedOutput*: La clase *ExpectedOutput* tiene como fin almacenar los datos o resultados esperados después de que se haya completado la ejecución del circuito cuántico. Este registro se contratará con el registro *ExpectedRegister*, que almacena el valor real de la ejecución del caso de prueba.
- Estereotipo *ValidationRegister*: La clase *ValidationRegister* se utiliza para validar y verificar el resultad obtenido del circuito principal con el resultado esperado predefinido, es decir, albergar los valores resultantes del matching entre el *ExpectedOutput* y el *ExpectedRegister*.
- Estereotipo *Qubit*: La clase *Qubit* almacena la unidad fundamental de información cuántica, pudiendo tener valor de 0 o 1. Esta unidad representa la versión cuántica de un bit clásico y puede estar en múltiples estados simultáneos gracias a la superposición cuántica. Representa tanto un cúbit de entrada, como un valor resultado.
- Estereotipo *Veredict*: Se añade la clase *Veredict*, siendo ésta la que proporcionará el resultado devuelto tras la ejecución de las pruebas del circuito cuántico, pudiendo ser el resultado Fail o Pass.
- Estereotipo *QuantumCircuit*: La clase *QuantumCircuit* será la encargada de almacenar el circuito cuántico sobre el cual se aplican las pruebas, también conocido como CUT o *Circuit Under Test*. La representación de un circuito, vendrá dada por una estructura de clases independiente del UMTP, por lo que esta Estereotipo constituirá el punto de anclaje entre el UMLTP y cualquier representación de un circuito cuántico basada en UML, como la que podemos encontrar en [10].

Tal y como se ha examinado previamente, un caso de prueba cuántico puede estar definido para circuitos deterministas (donde hay que realizar unas comprobaciones con respecto al resultado), o para circuitos de naturaleza estocástica (en los que habría que realizar comprobaciones probabilísticas que afectarían a cómo se representan los valores esperados y los mecanismos de comprobación del valor esperado con el obtenido).

Por ello, se añade el nuevo estereotipo (ver Figura V), *QuantumNondeterministicTestCase*, que representa al concepto de "Caso de prueba" para circuitos cuánticos no deterministas. Dada la escasez de propuestas en este sentido, esta clase queda de momento como un estereotipo representando el concepto, que más adelante habrá que especializar según las características de dicho tipo de prueba.

4.3. Extensión de la Vista "*Test Architecture Overview*". - En esta vista, se representan los diferentes componentes y módulos de pruebas, incluyendo casos de prueba, acciones de prueba, resultados esperados y otros elementos relacionados con la prueba. Estos componentes se organizan de acuerdo con la estructura de la arquitectura del sistema o componente que se está probando.

Además, la vista "*Test Architecture Overview*" muestra las relaciones entre los componentes de prueba y los elementos del sistema, tal y como pueden ser los componentes funcionales o las clases de diseño. Estas relaciones pueden incluir dependencias, asociaciones o herencias que indican cómo los componentes de pruebas están relacionados con los elementos del sistema que se están probando.

Por otro lado, esta vista es necesaria para describir cómo se estructuran las pruebas en relación con la arquitectura del sistema. Proporciona una representación visual de alto nivel que facilita la compresión de su organización (estructura del sistema bajo prueba) y relación entre los componentes de pruebas. Esta vista sirve de soporte a los equipos de desarrollo y pruebas para adquirir una visión clara de la arquitectura de las pruebas y de cómo se alinea con la arquitectura del sistema bajo pruebas.

A continuación, se muestra la Figura VIII en la que se pueden observar las partes en las que se organiza, a alto nivel, la vista "*Test Architecture Overview*".

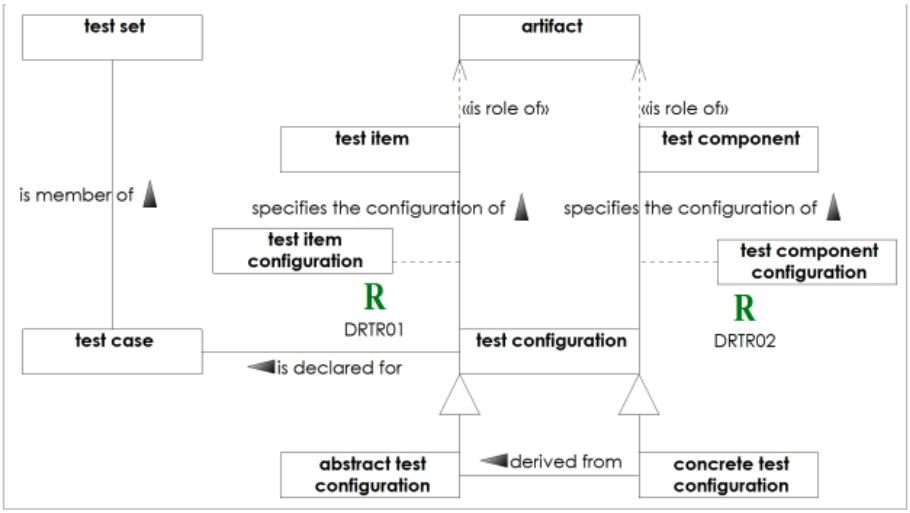


Figura VIII. Test Architecture Overview.

La extensión de la vista "*Test Architecture Overview*", que incluye los nuevos estereotipos que permiten dar cobertura a los conceptos sobre testing de software cuántico, puede observar en la Figura IX.

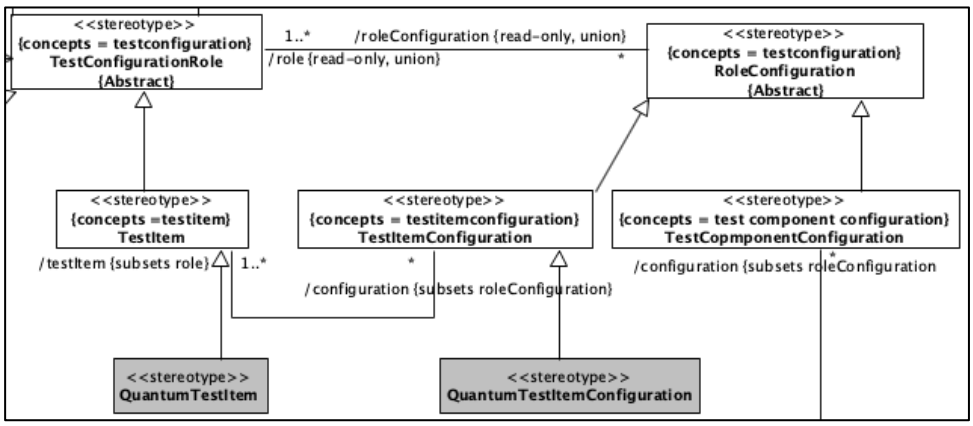


Figura IX. Fragmento del diagrama de clases de "Test Architecture Overview"

En el caso de esta vista, como se observa en la Figura IX, se observan dos nuevos estereotipos que se proponen para ser añadidos: *QuantumTestItem* y *QuantumTestItemConfiguration*.

El estereotipo *QuantumTestItem*, que hereda de *TestItem*, representa al artefacto bajo prueba que será objeto de la ejecución de los casos de prueba considerados dentro de la *TestConfiguration*. El *QuantumTestItem*, es lo que se conocería como *Circuit under Test* (o CUT). Puede considerarse como un circuito, una parte de un circuito cuántico, o un oráculo cuántico que va a ser sometido a prueba, y del que se espera obtener un veredicto en cuanto a su funcionamiento con respecto a unos casos de prueba cuánticos.

En el contexto de esta extensión, conviene mantener tanto el concepto general de *TestItem*, como el de *QuantumTestItem*, ya que los sistemas híbridos presentarán artefactos de ambos tipos, y esto implicará que, tanto en las pruebas funcionales como de integración, consideraremos artefactos de ambos tipos.

El estereotipo *TestItemConfiguration* representa al conjunto de restricciones que pueden ser aplicables al *TestItem* al que está asociado, y que constituye el sistema bajo prueba. Así pues, la extensión denotada por el estereotipo *QuantumTestItemConfiguration* (que hereda de *TestItemConfiguration*), haría referencia a restricciones específicas de un *QuantumTestItem*.

4.4. Extensión de la vista "*Test Case Overview*". - La vista "*Test Case Overview*" tiene como objetivo poder representar los casos de prueba para un sistema o componente software bajo prueba. En la vista "*Test Case Overview*", se identifican y organizan los casos de prueba, cada uno de los cuales se representa como un elemento individual en el modelo. Se puede incluir información como el nombre del caso de prueba, la descripción, los criterios de aceptación y los resultados esperados, sin incluir detalles de implementación del mismo.

Esta vista también muestra las relaciones y dependencias entre los casos de prueba y otros elementos del sistema, como los requisitos, las funcionalidades, los actores o los componentes específicos que están siendo probados. Estas relaciones permiten comprender cómo los casos de prueba se relacionan con los elementos del sistema y cómo se cubren los diferentes aspectos y escenarios de prueba.

Además, la vista "*Test Case Overview*" permite mostrar la agrupación de casos de prueba en "*test suites*" o grupos lógicos, organizando los casos de prueba en conjuntos coherentes y facilitando así la planificación y ejecución de las pruebas.

Por otro lado, esta vista proporciona una visión general de los casos de prueba, sus relaciones y su cobertura en el contexto del sistema o componente bajo prueba. Esto ayuda a los equipos de desarrollo y pruebas a comprender qué se está probando, cómo se están cubriendo diferentes aspectos y escenarios, y cómo se relacionan los casos de prueba con otros elementos del sistema.

En el contexto de esta vista, un *QuantumTestItem* podría tener asociadas distintas *TestItemConfiguration* siempre y cuando estas restricciones puedan ser aplicables a un software clásico. No obstante, si hubiera que añadir restricciones que sólo pudieran ser aplicables a un sistema cuántico, éstas deberían modelarse mediante el nuevo estereotipo *QuantumTestItemConfiguration*. Estas restricciones pueden hacer referencia a un lenguaje específico, un entorno de ejecución concreto, restricciones sobre qué tipo de servicio de ejecución puede ser válido (o no), etc.

A continuación, se muestra la Figura X en la que se puede observar las partes en las que se divide la vista "Test Case Overview" a alto nivel.

La extensión de la vista "Test Case Overview", que incluye los nuevos estereotipos que permiten dar cobertura a los nuevos conceptos, se puede observar en la Figura XI.

A continuación, se describen los estereotipos que representan los nuevos conceptos a considerar dentro de la vista "*Test Case Overview*".

El concepto de "caso de prueba cuántico" es tan novedoso que todavía no se ha establecido una definición aceptada y estandarizada para el mismo. Por este motivo, se decide considerar como nuevo estereotipo la clase *QuantumTestCase*, como especialización del concepto clásico "*TestCase*", pero preparada para recoger las particularidades del concepto que se consensue como caso de prueba para software cuántico. Este nuevo estereotipo, aunque será susceptible de ser completado en el futuro, considera por el momento el que se espera que sea una variable tipo entero, necesaria a especificar en todos los casos de prueba para software cuántico: *shots*. Esta variable representa el número de veces que debe ejecutarse el *QuantumTestCase* para poder obtener un resultado fiable y representativo para el mismo. Este número variará en función del servicio de ejecución cuántico que se emplee para testear el sistema bajo prueba.

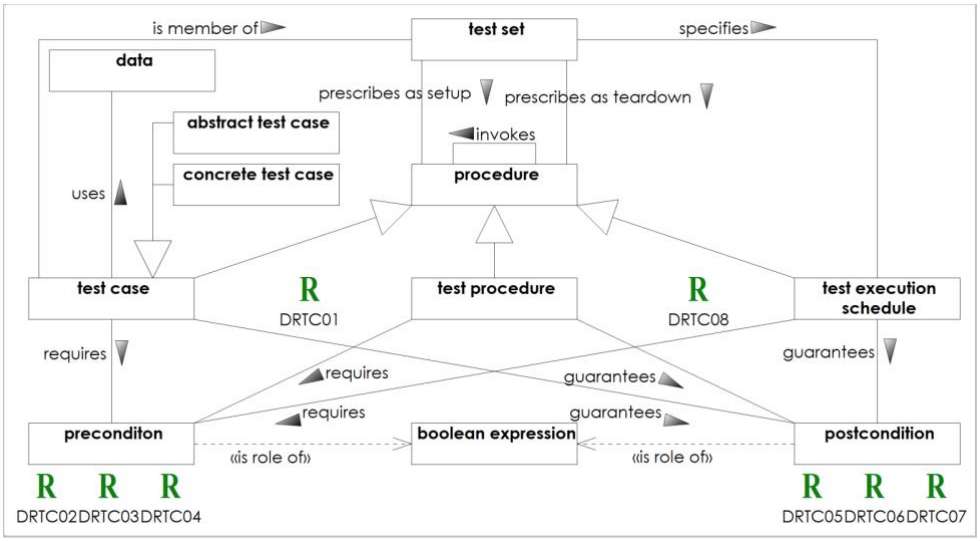


Figura X. Test Case Overview.

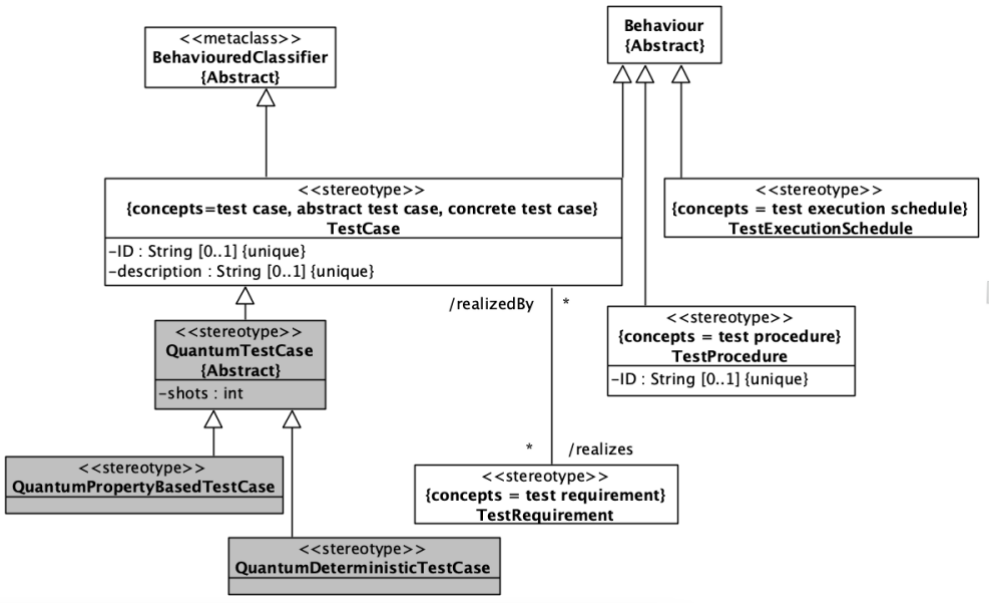


Figura XI. Diagrama de clases de "Test Case Overview".

Los tipos de circuitos cuántico que revela por ahora la literatura, en base a la naturaleza del resultado esperado, son dos: circuitos de comportamiento determinista o circuitos de comportamiento estocástico. El testeo de cada uno de estos circuitos requiere de mecanismos distintos a considerar, por lo que resulta importante considerar este hecho a la hora de reflejar los distintos tipos de prueba que pueden ser aplicables. Esto se materializa en las especializaciones *QuantumDeterministicTestCase* (elemento de modelado para casos de prueba de circuitos deterministas) y *QuantumPropertyBasedTestCase* (elemento de modelado para casos de prueba de circuitos no deterministas).

Según la literatura, se van encontrando enfoques de testing orientados a la Ingeniería del Software Cuántico en los que se plantea el proceso de testing de software cuántico según la naturaleza del circuito. Uno de estos enfoques responde a los circuitos deterministas. Los circuitos deterministas son aquellos en los que se obtiene un estado concreto como resultado de la ejecución del algoritmo codificado en el circuito cuántico. El caso de prueba, por lo tanto, debe realizar las tareas apropiadas para (i) considerar un valor esperado, (ii) gestionar la ejecución del CuT, y (iii) realizar las comparaciones necesarias para determinar en qué medida el resultado obtenido es igual al esperado.

El concepto que nos permite modelar este concepto es el *QuantumDeterministicTestCase*, y representa a este tipo de casos de prueba. En [1] encontramos un caso de caso de prueba para circuitos deterministas donde se consideran como parte integrante los elementos básicos de un caso de prueba clásico, tal y como se observa en la Figura XII.

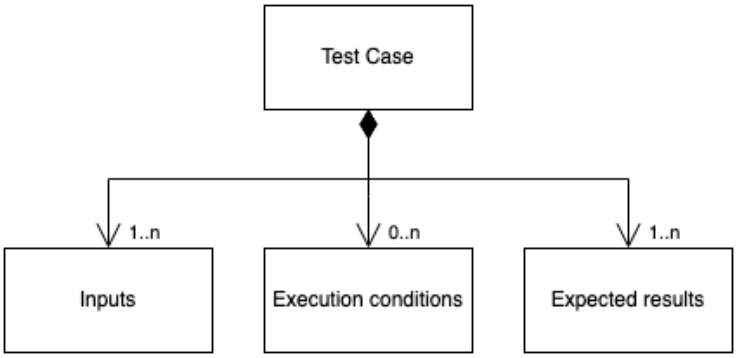


Figura XII. Partes de un caso de prueba [7].

Así, el caso de prueba cuántico en base a la propuesta de [4], se materializaría en un circuito de prueba (denominado *Quantum Test Circuit*, ver Figura XIII), que contemplaría no sólo los aspectos mencionados en la Figura XII, sino además, el propio CuT y los recursos necesarios para la evaluación del resultado. No obstante, esta definición tan precisa quedaría fuera de su definición en esta extensión, ya que podría considerarse como una implementación (aunque a nivel de modelo), concreta para un caso de prueba cuántico.

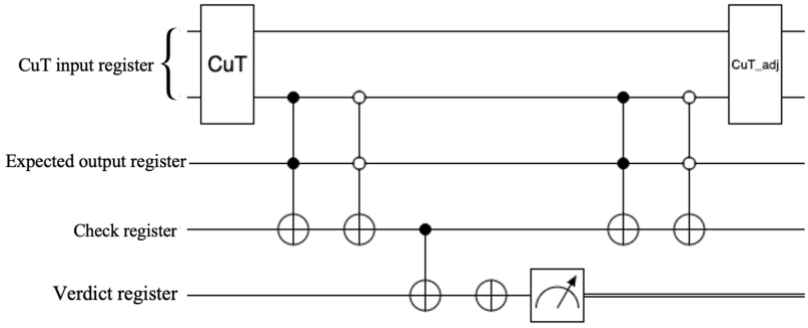


Figura XIII. Propuesta de Quantum Test Circuit para circuitos con comportamiento determinista.

Los circuitos cuánticos con un comportamiento no determinista requerirán de mecanismos adicionales para su prueba, teniendo en consideración que la evaluación del circuito puede conllevar analizar resultados que expresan distribuciones de probabilidades. Algunas propuestas recientes muestran cómo un circuito cuántico no determinista podría tener un comportamiento que debería evaluarse en base a propiedades del propio circuito. Las propiedades que podrían ser interesantes de evaluar en un punto concreto del circuito por ejemplo son: (i) Distribución de probabilidades; (ii) Valor determinista (clásico); o (iii) Estado de entrelazamiento.

Dichas propiedades, podrían modelarse tal y como se observa en la Figura XIV, sin embargo, podrían modelarse tantas como circunstancias o restricciones puedan identificarse en un circuito cuántico. Más allá de la propia propiedad con los datos básicos que se observan en la Figura XIV, el resto de información requerida para generar el caso de prueba sería relevante en tiempo de generación de código, por lo que no aplicaría como parte de la extensión del UMLTP.

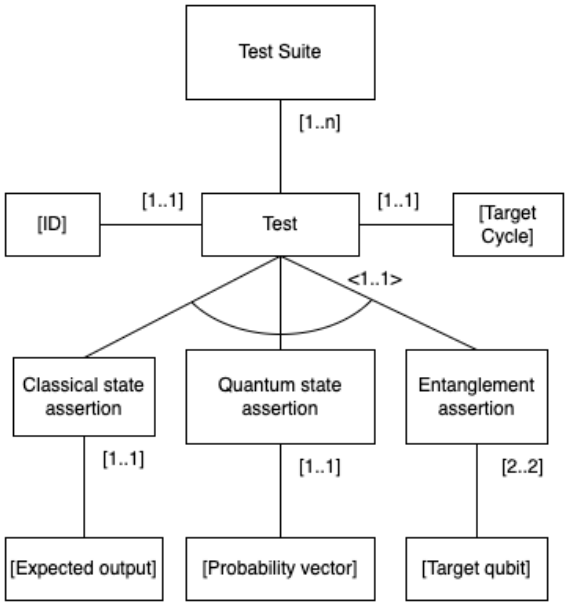


Figura XIV. Familia de tipos de prueba en base a las propiedades evaluables en un circuito.

4.5. Extensión de la vista "Arbitration and Veredict Overview". - La vista "*Arbitration and Veredict Overview*" del UMLTP proporciona una visión general de cómo se lleva a cabo el proceso de arbitraje y toma de decisiones durante la ejecución del test *suite*. Estas clases, gestionan también cómo se lleva a cabo la emisión del veredicto final en el contexto de las pruebas de software. Esto ayuda a garantizar la calidad del software al validar y confirmar los resultados de las pruebas realizadas.

Esta vista abarca los siguientes aspectos relevantes:

1. Arbitraje: Se refiere al proceso de resolver conflictos o discrepancias que puedan surgir durante la ejecución de las pruebas. Puede involucrar la comparación de resultados esperados y obtenidos, la revisión de requisitos o criterios de aceptación, y la toma de decisiones sobre la validez de los resultados de las pruebas.
2. Veredicto: Es el resultado o conclusión final que se obtiene después de completar las pruebas. El veredicto se basa en la evaluación de los resultados de las pruebas y puede indicar si se cumplen los criterios de aceptación, si se han encontrados errores críticos, si el sistema cumple con los requisitos establecidos, entre otros aspectos relevantes. Dado que un caso de prueba cuántico realiza por sí mismo la comparación del resultado obtenido con respecto al esperado, siendo este *fail* o *pass*, los veredictos expresados por los estereotipos de este paquete provienen de la propia ejecución de los casos de prueba cuánticos.

La Figura XV muestra, a alto nivel, las partes en las que se divide la vista "*Arbitration and Veredict Overview*".

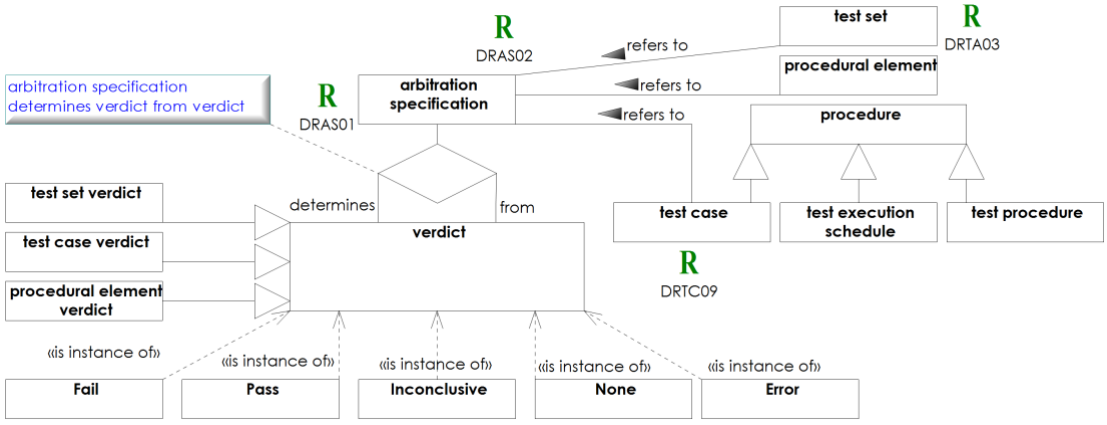


Figura XV. Arbitration & Verdict Overview

De acuerdo al análisis realizado sobre el estándar ISO/IEC 29119 y los estereotipos del UMLTP, no resulta necesario crear ninguna especialización de dichos estereotipos, pues los aspectos relativos al veredicto de la ejecución de la *test suite cuántica* (y con ello, sus *quantum test cases*), queda reflejado en la propia definición de los *quantum test cases*, siendo relevante únicamente si el veredicto ha sido uno de los establecidos como subclases del estereotipo *verdict* de la Figura XV.

5. Conclusiones. - La ingeniería dirigida por modelos presenta grandes ventajas a la hora de abordar la descripción agnóstica de sistemas complejos. Esta propiedad es de vital importancia para la QSE, dado que es uno de sus principios básicos y uno de los mecanismos mediante los cuales, se permitirá un desarrollo sostenible de software cuántico independiente de los lenguajes y plataformas que están apareciendo. Este principio debe aplicarse a todos los procesos de la QSE, siendo el testing de software cuántico el proceso en el que se enfoca este informe.

Otro aspecto crítico a la hora de abordar un enfoque agnóstico reside en que los artefactos que den soporte a la ejecución de un proceso de ingeniería del software puedan dar cobertura a la representación de todos los elementos relevantes en el modelado de dicho sistema cuántico (cúbits, puertas, oráculos, registros clásicos, etc.) en el contexto del proceso. Por ello, en el testing del software cuántico, la forma de asegurar esta cobertura de conceptos es la revisión de (i) el estándar por excelencia para el modelado de pruebas del software, es decir, UML Testing Profile, y (ii) la revisión del estándar ISO/IEC 29119, cuya definición de conceptos es la base para la creación del UML Testing Profile.

Partiendo de estos dos recursos, en este artículo se presenta un conjunto de estereotipos mediante los cuales, dar cobertura al modelado de pruebas para software cuántico. Dichos estereotipos, serán la base para la creación de entornos tecnológicos que permitan representar los artefactos de prueba para software cuántico de una forma independiente del lenguaje y la plataforma, dando así la oportunidad de definir las pruebas tan pronto como sea posible, y no sólo una vez que el software cuántico esté desarrollado. Seguiremos así las buenas prácticas que la comunidad industrial y académica han aprendido a lo largo de las últimas décadas, que se podrán aplicar al desarrollo de sistemas cuánticos especialmente críticos en cuanto a su calidad.

Agradecimientos. - Proyecto QSERV-UCLM (PID2021-124054OB-C32) financiado por el Ministerio español de Ciencia e Innovación (MICINN) y la Unión Europea. Ayudas para la realización de proyectos de investigación aplicada, en el marco del Plan Propio de Investigación, cofinanciadas en un 85% por el Fondo Europeo de Desarrollo Regional (Feder) UNION (2022-GRIN-34110).

Referencias

[1] Amo, A., M. Serrano, I. Guzmán, M. Usaola, and M. Piattini, Automatic Generation of Testing Circuits for Deterministic Quantum Algorithms. 2023.

[2] Firesmith, D. A Taxonomy of Testing. Software Engineering Institute's Insights (blog) 2015 Accesed at 02/04/2024]; Available from: https://insights.sei.cmu.edu/blog/a-taxonomy-of-testing/.

[3] García de la Barrera, A., I. García-Rodríguez de Guzmán, M. Polo, and M. Piattini, Quantum software testing: State of the art. Journal of Software: Evolution and Process, 2023. 35(4): p. e2419.

[4] García de la Barrera Amo, A., M.A. Serrano, I. García-Rodríguez Guzmán, M. Polo, and M. Piattini, Automatic generation of test circuits for the verification of Quantum deterministic algorithms. 2022. p. 1-6.

[5] Huo, Q., H. Zhu, and S. Greenwood, A Multi-Agent Software Environment for Testing Web-Based Application, in CMPSAC 2003. 2003, IEEE.

[6] Hutchinson, J., M. Rouncefield, and J. Whittle. Model-driven engineering practices in industry. in 2011 33rd International Conference on Software Engineering (ICSE). 2011.

[7] ISO/IEC, Software and system engineering - Software Testing - Part 1: Concepts and definitions. 2021.

[8] ISO/IEC, ISO/IEC/IEEE 29119-1:2022. Software and systems engineering. Software testing. 2022.

[9] OMG, UML Testing Profile 2. 2019, Object Management Group.

[10] Pérez-Castillo, R., L. Jiménez-Navajas, and M. Piattini, Modelling Quantum Circuits with UML. 2021. p. 7-12.

[11] Piattini, M., G. Peterssen, and R. Pérez-Castillo, Quantum Computing: a new Software Engineering golden age. ACM SIGSOFT Software Engineering Notes, 2020. 45(3): p. 12-14.

[12] Piattini, M., 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. 2020.

[13] Piattini, M., M. Serrano, R. Perez-Castillo, G. Petersen, and J.L. Hevia, Toward a Quantum Software Engineering. IT Professional, 2021. 23(1): p. 62-66.

[14] Polo, M., I. García-Rodríguez de Guzmán, A. García, M.Á. Serrano, M. Piattini, A. Martínez, and G. Peterssen, Chapter 7: Quantum Software Testing, in Quantum Software Engineering & QuantumPath®, G. Peterssen, J.L. Hevia, and M. Piattini, Editors. 2023, aQuantum. p. 372.

[15] Shaukat Dar, K., U. Shaukat, F. Feroz, S. Kayani, and A. Akbar, Taxonomy of Automated Software Testing Tools. International Journal of Computer Science and Innovation, 2015. 1: p. 7-18.

[16] Villalón, J.C.M., G.C. Agustin, T.S.F. Gilabert, and J.d.J.J. Puello. A taxonomy for software testing projects. in 2015 10th Iberian Conference on Information Systems and Technologies (CISTI). 2015.

Nota contribución de los autores:

1. Concepción y diseño del estudio
2. Adquisición de datos
3. Análisis de datos
4. Discusión de los resultados
5. Redacción del manuscrito
6. Aprobación de la versión final del manuscrito

IGRG ha contribuido en: 1, 2, 3, 4, 5 y 6.
MRM ha contribuido en: 1, 2, 3, 4, 5 y 6.
MPV ha contribuido en: 1, 2, 3, 4, 5 y 6.
MTMQ ha contribuido en: 1, 2, 3, 4, 5 y 6.

Nota de aceptación: Este artículo fue aprobado por los editores de la revista Dr. Rafael Sotelo y Mag. Ing. Fernando A. Hernández Gobertti.