Dissertation

Supervisor: Carlos Perez Delgado
Programme: MSc Computer Science (Artificial Intelligence)
Word Count: 10,000

December 3, 2024

Acknowledgements

Abstract

List of Figures

List of Tables

List of Abbreviations

- AI - Artificial Intelligence
- MSc - Master of Science

Contents

1 Introduction

1.1 Problem Description and Incentive

Quantum Software Engineering (QSE) is an emerging research field seeking to develop and standardise software engineering principles in quantum technologies. Many of these standards have been adapted from classical software engineering (CSE), attempting to keep them as familiar as possible to CSE whilst being able to distinguish that quantum and classical systems are two fundamentally different hardware.

One area of interest in QSE is the adaptation of Unified Modelling Language (UML) to model quantum systems and illustrate their communication with classical systems. Two notable papers have introduced adaptations of UML for this purpose: Q-UML [5] and a quantum UML profile [4]. Both propose methodologies for incorporating quantum technologies into UML, aiming to broaden the scope of professionals who can contribute to the QSE field and promote early adoption of a standardised quantum modelling language whilst quantum technologies remain in relative infancy.

The critical question is which quantum UML adaptation offers the best solution. This involves evaluating their effectiveness for modelling quantum systems, their suitability in real-world applications and their potential for widespread adoption. Additionally, should the industry favour full-scale UML modelling or a simplified approach, such as flowcharts, for quantum system design, and which of the two adaptations can best accommodate these design preferences?

1.2 Goals and Objectives

This project aims to create and contrast UML diagrams using these two quantum UML approaches.

The first objective is to choose an appropriate real-world example as a starting point for the initial diagrams. The example chosen is the Variational Quantum Eigensolver (VQE), a hybrid quantum/classical algorithm. VQE is ideal for exploring how communication between quantum and classical machines can be represented in a modelling language.

The second objective is to choose the most suitable UML diagram to model the algorithm and compare the quantum UML methods. A sequence diagram was selected because it is one of the most commonly used UML diagrams and illustrates the communication between quantum and classical modules throughout the VQE algorithm.

The third objective is to determine a robust method for comparison and analysis. This will involve combining the author's observations, following established guidance on good diagram design, and feedback gathered from anonymous polling. Quantum computing and software engineering professionals would be ideal participants in this feedback process.

Finally, the last objective is to create multiple UML diagrams. A valuable exploration will involve modelling fundamental quantum properties such

as entanglement and superposition, as proposed by the authors of the quantum UML literature. This approach will enable the use of other UML diagrams or the application of sequence diagrams in a new context.

2   Literature Review

2.1   Background

2.1.1   Q-SE 2020

Q-SE 2020 was the first international workshop held virtually in July 2020, co-located with ICSE 2020. Its purpose was to unite researchers and professionals to form a community that could advance quantum software development.

Paper submissions were requested, including "Towards a Quantum Software Modelling Langauge" [6], which laid the groundwork for Q-UML, which was further expanded upon in the paper "A Quantum Software Modeling Language", presented in "Quantum Software Engineering" [7].

The authors of "Quantum Software Engineering" [7] developed their own quantum UML profile, first published in March 2021 [3], which was later expanded on and presented in "Computing" [4].

Both papers sought to adapt UML to integrate quantum computing principles.

2.1.2   UML 2

UML 2, simply referred to as UML, is a modelling language and the industry standard for designing and documenting computer systems. It includes 14 types of diagrams divided into structural and behavioural categories. UML 2 signifies that we are currently using the second version of this language.

Sequence diagrams, part of the behavioural category, belong to a sub-category of interaction diagrams, which includes communication, timing, and interaction overview diagrams.

Sequence diagrams model communication protocols between human and non-human entities. The horizontal axis represents the sequence of communication messages, while the vertical axis indicates the timing of interactions. Each component in a sequence diagram has a lifeline extending vertically, which may terminate if the element is no longer involved in the sequence.

As the industry standard for CSE, UML is the logical choice for adaptation for QSE. To evaluate its effectiveness, it should be tested and modelled using current quantum hardware.

2.1.3   NISQ

The Noisy Intermediate Scale Quantum (NISQ), known as the near-term quantum computer, employs qubits ranging from tens to hundreds. An NISQ device

is sensitive to noise and requires error correction. We are currently in the NISQ era, which means quantum advantage has yet to be achieved.

Despite this, computations can still be performed on NISQ devices, with workloads typically split between classical and quantum computers. One such computation that can be accomplished through this hybrid approach is the Variational Quantum Eigensolver.

2.1.4 VQE

The Variational Quantum Eigensolver (VQE) is a hybrid algorithm that leverages both quantum and classical devices to find the ground state of a given physical system.

The algorithm starts with the physical system represented as a Hamiltonian and prepares a trial state called the ansatz, which is parameterised. A quantum computer evaluates the ansatz by applying a quantum circuit to explore the parameter search space. A classical optimiser then adjusts the parameters in each iteration, guiding the quantum circuit's search and gradually refining the ansatz until the algorithm converges on the lowest energy estimate, the ground state of the physical system.

2.2 Related Work

2.2.1 Q-UML

Q-UML is a methodology for integrating quantum computing into the UML standard. The paper establishes a fundamental axiom and five core design principles to consider when designing a UML quantum extension. As the axiom asserts, while the design should adhere to classical standards as closely as possible, it must acknowledge the fundamental differences between classical and quantum hardware. The design principles describe when and how to label components in a diagram as classical or quantum.

In Q-UML, quantum elements are visually distinguished using bold font for textual labels and double lines for diagrammatic components.

2.2.2 Quantum UML Profile

A quantum UML profile is an extension of UML designed explicitly for hybrid information systems incorporating classical and quantum software.

In UML, a profile is a structural diagram that extends the language to support domain-specific custom models. The quantum UML profile introduces a set of stereotypes, tagged values and constraints to represent quantum concepts.

3   Design and Implementation

3.1   Plant UML

The initial version of the VQE sequence diagram was drafted using the Plant UML plugin for PyCharm, an open-source tool that enables users to generate UML diagrams from plain text.

To create the Q-UML version of the VQE sequence diagram, a combination of Plant UML and Lucidchart was used to include the double lines and bold text necessary to distinguish quantum components. However, for creating the quantum UML profile, Plant UML alone may suffice, as it should have the tools required to extend UML to represent quantum-specific elements.

3.2   Lucidchart

Lucidchart is a web-based application that creates general-purpose diagrams, including UML diagrams, through a graphical user interface (GUI). It is used when creating Q-UML adaptations as it allows the customisation of elements to contain bold text and double lines necessary to distinguish quantum components.

3.3   Qiskit

Qiskit is an open-source software development kit (SDK) created by IBM to access and utilise their cloud-based quantum computing services. Implemented in Python, Qiskit provides tools and libraries for quantum programming and experimentation. The VQE algorithm represented in the sequence diagram is based on the implementation of VQE in Qiskit [2], which interacts directly with the IBM Quantum Platform.

3.3.1   Qiskit and VQE

3.4   Design of VQE & QUML

This diagram has been completed and is attached at the end of this report. It illustrates the sequence of messages between modules necessary to execute the VQE algorithm.

Multiple Pauli operators define the Hamiltonian "sparsely" instead of creating a complete matrix. It acts upon two qubits at a time, and the qubits count is passed to the ansatz object. The ansatz is a circuit that creates a trial state for the experiment.

These objects are transformed through a pass manager to be compatible with a quantum computer, the "backend". This process does not require communication with the quantum hardware, as the information is held classically; therefore, the diagram does not depict these modules as quantum.

A session is created and configured to the same backend, passing multiple circuits to an estimator object while the session handles resource management.

This estimator is a quantum object, depicted as such in the diagram, as it utilises quantum hardware to estimate the values of the quantum circuits.

The methods `minimize` and `cost_func` contain the estimator object and are also depicted as quantum. The `cost_func` takes the Hamiltonian, ansatz, and parameters (initially a NumPy array of random values) and combines them into a new variable, `pub`, which is then given to the estimator object. The estimator executes 10,000 shots, defined during its creation, meaning the quantum circuit runs 10,000 times. This message is marked as a quantum message with **bold text**, indicating that the communication uses quantum hardware. The estimator then returns an energy estimate, converted to classical information before being received by the `cost_func`, therefore depicted as a classical message.

The `cost_func` method updates the parameters and records critical information in a `Dictionary` object per iteration. The `minimize` method, acting as an outer loop, receives the energy estimate and uses a classical optimiser (COBYLA) to guide the process into finding the lowest energy estimate.

The loop is completed once the energy estimate returned by `cost_func` converges to the lowest energy estimate, and the parameters and iterations are verified with the `Dictionary` object. The final step involves plotting each iteration and energy estimate using the Matplotlib Python library.

3.5 Design of VQE & Quantum UML Profile

This has yet to be completed and will be the next step in the process.

3.6 Design Visualisation

The first completed diagram is a PNG file created in Lucidchart. An attempt was made to keep it to a format similar to the Plant UML output. Consistency must be considered to ensure a fair comparison between both diagrams; therefore, even if the VQE quantum UML profile diagram can be reproduced in Plant UML alone, it may need to be translated into Lucidchart to ensure this consistency is upheld.

4 Results and Analysis

4.1 Author Observations

My initial observations of the first diagram suggest that while the bold text and double lines used to distinguish quantum components are visible, they are more easily identifiable in examples provided in the Q-UML paper. This is likely due to the larger scope of the VQE sequence diagram and the fact that it is primarily composed of classical elements. Additional design choices could be considered to better differentiate between classical and quantum components, although these would fall outside UML standards. For example, different colours could represent classical and quantum modules. This feature would only be readily available in applications like Lucidchart but could be incorporated with

Plant UML by developing an extension. Once the quantum UML profile version of the VQE sequence diagram is completed, more detailed observations and comparisons between the two diagrams will be possible.

4.2 Diagram Literature

Once both diagrams are completed, it will be crucial to consult relevant literature on effective diagram design to make an informed assessment. Research papers exploring good diagram design, particularly concerning UML, should be considered. An example of such work is the paper "Improving Information System Design: Using UML and Axiomatic Design" [1], which provides valuable insights into improving system information design.

4.3 Polling

An anonymous questionnaire should be developed to present both VQE sequence diagrams to professionals and researchers in quantum computing and software engineering. Ideally, multiple UML diagrams would be included. The questionnaire should ask participants about the clarity of the diagrams, their understanding of the content being depicted, and their overall observations on the application of UML to quantum technologies. Additionally, gathering information on the industry sectors respondents are associated with would be valuable, providing insight into the diversity of perspectives within the results.

5 Conclusions

The paper will conclude with an overall assessment of the advantages and disadvantages of each method. It will address whether the primary goal and all four main objectives were achieved. Additionally, there will be a discussion on whether UML is the most suitable approach for representing quantum systems or if a more general flowchart format might be more appropriate. This will include considering which method could be used in such a case. I assume that Q-UML would remain the most applicable, as the quantum UML profile is inherently based on UML formatting.

6 Future Work

Depending on whether the fourth objective is achieved, further exploration into modelling fundamental quantum concepts would be beneficial for evaluating the effectiveness of applying UML to quantum systems. Plant UML is currently the most convenient tool for creating UML diagrams, as Lucidchart is significantly more labour-intensive. Developing an extension for Plant UML or other open-source tools like Mermaid to incorporate Q-UML formatting would be highly advantageous for future diagram creation. Such an extension would also increase

the utility of this method, making it more accessible for continued exploration and broader adoption in the QSE field.

References

7 Bibliography

References

[1] Luís Cavique et al. "Improving information system design: Using UML and axiomatic design". In: *Computers in Industry* 135 (2022), p. 103569. ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2021.103569. URL: https://www.sciencedirect.com/science/article/pii/S0166361521001767.

[2] IBM Quantum Learning. *Variational Quantum Eigensolver*. https://learning.quantum.ibm.com/tutorial/variational-quantum-eigensolver. Accessed: 5th October 2024. 2024.

[3] Ricardo Pérez-Castillo, Luis Jiménez-Navajas, and Mario Piattini. "Modelling Quantum Circuits with UML". In: *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*. 2021, pp. 7–12. DOI: 10.1109/Q-SE52541.2021.00009.

[4] Ricardo Pérez-Castillo and Mario Piattini. "Design of classical-quantum systems with UML". In: *Computing* 104.11 (2022), pp. 2375–2403. ISSN: 1436-5057. DOI: 10.1007/s00607-022-01091-4. URL: https://doi.org/10.1007/s00607-022-01091-4.

[5] Carlos A. Pérez-Delgado. "A Quantum Software Modeling Language". In: *Quantum Software Engineering*. Ed. by Manuel A. Serrano, Ricardo Pérez-Castillo, and Mario Piattini. Cham: Springer International Publishing, 2022, pp. 103–119. ISBN: 978-3-031-05324-5. DOI: 10.1007/978-3-031-05324-5_6. URL: https://doi.org/10.1007/978-3-031-05324-5_6.

[6] Carlos A. Pérez-Delgado and Hector G. Pérez-Gonzalez. "Towards a Quantum Software Modeling Language". In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. IEEE, 2020, pp. 442–444. DOI: 10.1109/ICSEW48907.2020.00073. URL: https://doi.org/10.1109/ICSEW48907.2020.00073.

[7] M.A. Serrano, R. Pérez-Castillo, and M. Piattini. *Quantum Software Engineering*. Springer International Publishing, 2022. ISBN: 978-3-031-05323-8. DOI: 10.1007/978-3-031-05324-5.

8 Appendix