# Exploratory Review of Quantum Computing Software Requirements Specification and their Measurement

Tuna Hacaloglu[1,2], Hassan Soubra[3], Pierre Bourque[1]

[1]École de Technologie Supérieure, 1100, rue Notre-Dame Ouest, Montréal, Québec, H3C 1K3, Canada
[2]Atilim University, İncek, Gölbaşı, Ankara, 06830, Türkiye
[3]École Centrale d'électronique (ECE), Lyon, France

**Abstract**
Quantum software sets itself apart from classical software owing to its powerful computational abilities rooted in entanglement and superposition. Unlike classical software, quantum software diverges notably across various dimensions, including computational models, hardware architectures, algorithms, deployment platforms, and problem domains. Quantum software is also often not standalone and interacts heavily with classical software, stressing the importance of carefully considering hybridization. From a software engineering standpoint, researchers generally agree that a different approach is required for quantum software, and they advocate a Quantum Software Development Life Cycle (SDLC). This exploratory study briefly outlines the specifics of quantum software, overviews the proposed approaches regarding the software requirements of quantum software, and then reviews the current alternatives for measuring the functional size of quantum software. This study indicates that only a few papers in the literature discuss the requirements and functional size measurements of quantum software. Their results are also mostly conceptual and have not yet been empirically validated. Functional size measurement using quantum software remains an open area for further research.

**Keywords**
Software size measurement, quantum software, quantum software requirements [1]

## 1. Introduction

The revolutionary potential of quantum computing arises from its unique capabilities, such as entanglement and superposition, and its promising unprecedented computational power. In this context, quantum software is a beacon for innovation. While classical software has long served as the cornerstone of digital innovation, quantum software is emerging as a distinct entity that offers a fundamentally different approach to computation. This distinction encompasses a multitude of facets ranging from hardware architectures to the algorithms employed, along with the platforms on which they are deployed and the problem domains they address.

This study explores the emerging field of quantum software from a measurement perspective. First, we introduce the features that distinguish quantum software from classical software, and the need for a development life cycle specific to quantum software. Next, we will focus on describing the software requirements for quantum software, which serves as the fundamental measurable input essential to the software development process. By examining how software requirements are addressed in quantum software, we identify similarities and differences with classical software requirements and strive to determine potential elements of improvements and research gaps that could serve as sources of change in the field of

IWSM-Mensura, September 30–04, 2024, Montréal, Canada
tuna.hacaloglu@etsmtl.ca (T. Hacaloglu); hsoubra@ece.fr (H. Soubra); pierre.bourque @etsmtl.ca (P. Bourque)

software measurement. Additionally, we introduce the methods recommended in the literature for quantum software size measurements and provide a comparison of these methods. This exploration will allow the development of insights into the evolving landscape of quantum software and offer guidance for researchers, practitioners, and stakeholders alike, as they navigate the complexities of this new technology from the software size measurement perspective.

The remainder of this paper is structured as follows: Section 2 presents a background related to quantum software development. Section 3 presents requirements engineering studies for quantum software. Section 4 presents studies on quantum software size measurements. Section 5 presents a discussion, and Section 6 concludes the paper.

## 2. Background

Quantum computers, which surpass the capabilities of classical computers, have the potential to provide more effective and innovative solutions for the technological needs of the future. Quantum computing applications extend to complex fields, such as drug discovery, radiotherapy optimization, and cryptography, each presenting multidimensional challenges that highlight the importance of Quantum Software Engineering (QSE) [1].

Quantum software (QS) differs from classical software in several respects. First, their computation models diverge from classical software that relies on bits, whereas QS operates with more complex quantum bits (qubits). Second, their hardware architectures differ significantly; classical software uses logic gates, whereas QS utilizes quantum gates. Additionally, the fields they target differ significantly: classical software caters to a wide variety of applications such as games, web applications, and databases, whereas QS is geared towards more computation-intensive applications such as optimization, cryptography, and simulation. Superposition and entanglement are features that distinguish quantum computing from classical software [1]. Owing to superposition, a quantum program can exist in multiple states simultaneously. Because of entanglement, two entangled qubits or registers exist in a single quantum state; therefore, two qubits can contain four values at once (e.g., 00, 01, 10, and 11) [1]. However, quantum noise and decoherence, which affect the precision and performance of quantum programs, are challenges that cannot be overlooked or easily addressed by the upcoming Noisy Intermediate-Scale Quantum NISQ computers [2], [3]. The authors in [1] pointed out that given the unique properties of quantum computing, such as superposition and entanglement, classical software engineering methods are inadequate for creating efficient quantum software applications, and there is a need to devise new Quantum Software Engineering (QSE) methodologies.

At this point, the consensus among researchers is that there is a need for a quantum Software Development Life Cycle (SDLC) and that it should be approached differently from classical software because of quantum-related challenges. Further support by the authors in [1], [4] emphasize the hybrid nature of quantum applications and draws attention to the lack of an SDLC approach that addresses the quantum-classical integrated application development challenges. In [5] the authors explained this hybrid nature of quantum applications as a quantum program that utilizes a quantum register consisting of qubits for executing quantum operations alongside a classical register containing classical bits to store observations of qubit states and conditionally apply quantum operators. Several approaches to SDLC have been presented in the literature [4], [6], [7]. However, an examination of these studies reveals a noticeable scarcity of research on the measurement of quantum software projects.

Software measurement plays an important role in moving forward efficiently and successfully during SDLC and is of great importance in quantum projects, just as it is in classical projects. More specifically, measurement contributes to the software engineering discipline through cost estimation, performance assessment, process improvement, decision making, and quality control. Software measurement, an important component of project management, must be conducted effectively to ensure proper management of quantum software development projects.

From a management perspective, software requirements, which are the fundamental reason for developing software, play a crucial role from a measurement perspective. It is crucial to define requirements in the early phases of software development to ensure that all stakeholder needs and specifications are accurately identified and documented [8]. The early identification of requirements is important not only to bridge the communication gaps among stakeholders but also to conduct software measurements for early estimation. Because Quantum Software engineering is still emerging as a new field, quantum software requirement studies are currently quite limited, with only a handful of studies investigating requirements in the quantum software development domain [6], [9], [10], [11], [12]. The existing literature offers some insights into the specific requirements of quantum software development. In [1] the authors draw attention to the challenges that quantum computing brings and posits that, like classical software, quantum software engineering demands the creation of innovative techniques for elicitation, specification, modeling, analysis, and verification. The authors also emphasize the need for further investigation to ascertain whether expanding UML would be adequate or if there is a need for more domain-specific modeling solutions.

In summary, quantum software development employs a hybrid methodology that blends elements from both quantum and classical software development methodologies, potentially influencing the abstraction and presentation of requirements and their management. For this reason, classical requirements engineering must be revisited to comply with quantum software development.

## 3. Advancements in Requirements for Quantum Software Development

Only a handful of studies have investigated the requirements of Quantum Software Development. Dey et al. [6] suggested dividing quantum requirement specifications into two categories: quantum software requirement specification and quantum hardware requirement specification. According to Dey et al. [6], the quantum software requirements specification encompasses quantum tools, integrator plugins, logical circuit synthesizers for quantum systems, classical validators, and classical software requirements specification modules, whereas quantum hardware requirements specifications include qubit count, quantum volume, physical machine description, and classical hardware processor. Dey et al. [6] highlighted the importance of having or developing a clear understanding and explicit declaration of hardware requirements before proceeding to the design phase. The authors in [6] also described quantum hardware specifications by pointing out that the performance of quantum computation can be impeded by various technological challenges such as inadequate qubit count, low qubit fidelity, qubit errors, and shorter coherence intervals.

Yue et al. [9] presented aspects that distinguished quantum requirements engineering from classical software requirements. They analyzed requirements engineering based on major concepts such as stakeholders, functional and non-functional requirements, and requirement specification, and compared classical requirements with quantum requirements. A summary of the proposal of Yue et al. [9] is presented in Table 1.

**Table 1:** Quantum requirements engineering in [9]

| Requirements engineering related components | Quantum software related suggestions in the paper |
|---|---|
| Stakeholders & System boundaries | Identifying stakeholders & system boundaries remain the same as in the classical software context. |

| | |
|---|---|
| Requirements Gathering | It is anticipated that established techniques like interviews and prototyping for gathering requirements will remain largely unchanged. |
| Requirements Specification | Specifications using modeling notations need to be updated for incorporating quantum aspects. Requirements specifications for quantum software will undergo alterations to incorporate concepts pertinent to quantum software. The authors designed a use case diagram to cover use cases in the classical domain, the quantum domain, and a hybrid of them. |
| Requirements Categorization | The requirements should be classified into the ones for the classical parts of the system and the other for the quantum parts of the system. Therefore, they define these requirements as classical, quantum and hybrid requirements. |
| Functional Requirements | Identifying functional requirements for quantum software is the same as for classical software. |
| Extra Functional (Non-Functional) Requirements | While all the non-functional requirements defined for classical software in SWEBOK [13] will be relevant to quantum software systems too, there needs to be additional requirements for quantum software. Non-Functional Requirements (NFR) specific to Quantum software are given in Table 2. |

Yue et al. [9] also drew attention to quantum-specific nonfunctional requirements. They are summarized in Table 2.

**Table 2:** Non-Functional Requirements (NFR) specific to Quantum software in [9].

| NFR | Specific importance in Quantum Software |
|---|---|
| Portability | Due to the potential development of quantum computers using different technologies, portability will be critical. |
| Performance | In terms of estimating the number of qubits and gates, performance requirements will be relevant in determining in advance whether the available resources can meet the anticipated performance criteria. |
| Reliability | Defining reliability requirements become particularly important when considering how hardware errors can affect the reliability of quantum software systems. |
| Scalability | Considering that quantum computers are known to support a limited number of resources (e.g., number of qubits, depth of the quantum circuit), it becomes crucial to define scalability requirements. |
| Maintainability | With the ongoing technological advancement of quantum hardware, it will be necessary to update existing quantum software to accommodate hardware changes. |
| Reusability | Quantum software, typically developed as a hybrid of classical and quantum components, faces reduced reusability due to strong interdependence between these components. Enhancing cohesion within the quantum component can improve its reusability. |

Yue et al. [9] also explored the modeling of quantum software using a UML use-case diagram. They used a credit risk analysis example using a quantum algorithm. In their use case diagram, there are actors from both the classical domain (credit analyst) and quantum domain (quantum expert). Similarly, the use cases in the diagram include classical requirements such as "determine the confidence level", and hybrid requirements such as "manage risk in finance with quantum" and purely quantum related such as "estimate the required number of gates" and "estimate the impact of hardware noises".

Saraiva et al. [10] suggested that non-functional requirements (NFR) are oriented toward hardware-related constraints of quantum computing. They also mapped these to the product quality characteristics of the ISO 25010 Quality Model [14] (see Table 3). These product quality characteristics are related to the performance efficiency, resource utilization, and reliability in the ISO 25010 Quality Model.

**Table 3:** Non-Functional Requirements Specific to Quantum Software in Saraiva et al. (2021)

| Quantum Software NFR | Equivalent product quality characteristic in [14] |
|---|---|
| [NFR1 - The program should use a maximum of n qubits, where n is the number of qubits available in the target quantum device.] | Performance efficiency<br>Resource utilization |
| [NFR2 - The program should be designed considering the maximum circuit depth so that the target device can maintain a stable quantum state for the necessary period to execute the algorithm.] | Reliability |
| [NFR3 - The program should be designed considering the number of T gates so that it does not exceed the limit of the target device.] | Reliability |
| [NFR4 - The program should be implemented minimizing the number of gates between qubits that are not physically connected on the target device.] | Performance efficiency<br>Reliability |
| [NFR5 - The program should be implemented minimizing the use of gates that are not available in the target quantum device.] | Performance efficiency<br>Reliability |

Moreover, the literature includes some endeavors aimed at representing quantum software. For instance:

- Perez-Delgado and Perez-Gonzalez [15] presented a modeling language for quantum software based on the Unified Modeling Language (UML), which includes a concise set of extensions to UML specifically tailored for modeling quantum software but can also be used separately and independently of UML. These extensions included class and sequence diagrams. In their example, they modelled Shor's algorithm by using extended class and sequence diagrams.
- Pérez-Castillo et al. [16] modeled a teleportation algorithm in a quantum circuit using a UML activity diagram with swim lanes. In their activity diagram, the flows between the qubits, gates, measures, and registers are represented.

## 4. Studies in Quantum Software Sizing

In terms of measurements, some methods have been proposed in the literature. Sicilia et al. [17] suggested a preliminary set of research directions for quantum software measurement, acknowledging that their recommendations are provisional and incomplete owing to the anticipated rapid evolution of the quantum software field.

Zhao proposed that the size of quantum software can be measured according to three major aspects and abstraction levels: code, design, and specification levels. For each of these aspects, the author suggests extending the classical measures to quantum software [18]. For instance, Zhao suggested measuring the software size using LOC metrics adapted to incorporate quantum-related code parts into the overall LOC. However, these concepts have not yet been empirically validated or tested [18].

To the best of our knowledge on functional size measurement (FSM) using Quantum Software, the literature includes only three studies [5], [19], [20]. All three studies were based on COSMIC – ISO 19761 [21], which is a functional size measurement method that measures the size of a given piece of software by counting the data movements within its functional requirements. These data movements were categorized as Entry (E), Exit (X), Read (R), and Write (W). The measurement unit of functional size was measured in the COSMIC Function Points (CFP).

In Table 4, we present a comparison of these three approaches in terms of the inputs used for measurement, functional processes, data movement types, and their respective units. However, the practicality of these studies in measuring the size of quantum software has not yet been investigated.

**Table 4:** Comparison of 3 studies on FSM approaches for sizing Quantum Software

| Article | Measurement Input | Functional process | Data movement types | Unit of Data movement |
|---|---|---|---|---|
| [5] | Qiskit software | Quantum Gate | E: each QUBIT connected via a line to a quantum gate; each gate connected via a line to a Functional process<br>X: each line to a Functional process<br>R: each read from a classical bit<br>W: each Quantum Measure identified in this FP | CFP |
| [20] | Use-case diagram with Q-UML | Use case | QE: Quantum Entry<br>QX: Quantum Exit<br>QR: Quantum Read<br>QW: Quantum Write | QCFP=CFP |
| [19] | Quantum circuits | The actions performed on qubits-input and operator actions of the circuit. | E<br>X<br>R<br>W | CFP |

## 5. Discussion

When examining studies of quantum software in the literature, it is evident that quantum software possesses unique characteristics that distinguish it from classical software. In the context of quantum software engineering, studies on requirements engineering and quantum software measurements are scarce and limited to conceptual research. However, these studies have not been validated or empirically tested, indicating that this field is still being developed. Another insight from the literature reveals that the definition of requirements scope lacks precision in the context of quantum software and hybrid software, with a parallel issue arising in the requirement specification. Researchers have outlined quantum-specific functional requirements [9] and quantum-specific nonfunctional requirements [9], [10]. Furthermore, Dey et al. [6] classified requirement specifications for quantum software into two distinct types: hardware requirements and software requirements specifications.

Two non-functional requirement (NFR) categories are commonly observed in studies: reliability and performance. Yue et al. [9] explored the impact of hardware errors on the reliability of quantum-software systems. A crucial aspect of the discourse on non-functional requirements involves addressing errors and noise correction tailored to quantum software. This prompts an inquiry into how to articulate them within the framework of NFR and whether they might evolve into functional requirements—an area ripe for further investigation.

Furthermore, the definition of the requirements for quantum software must be reconsidered because quantum software possesses complex characteristics [1], [4], [5] . Classical software systems are often perceived as black boxes that demonstrate the capabilities of the actors within the system, typically encompassing use cases. This reassessment is necessary to include the quantum software requirements. For instance, Yue et al. [9] defines hardware constraints specifically tailored for quantum software as additional functional requirements.

In addition, quantum computers are inherently limited in their capacity. The quantities of qubits and gates are of particular significance in the development of quantum software. Yue et al. [9] highlighted their significance within the domain of performance requirements, stating that in estimating the number of qubits and gates, performance requirements become pertinent for preemptively assessing whether the available resources align with the envisaged performance [9]. Sicilia et al. [17] highlighted the existence of various "quantum instruction sets" designed to translate algorithms into physical instructions. These "quantum instruction sets" provide a programming experience like that of assembly or virtual machine programming, often tailored for specific hardware platforms.

Another open question for research is how the measurement of size should be defined for quantum software: whether it should be defined functionally or whether the criteria of size measurement should be broadened to include attributes specific to quantum software.

Darwish and Soubra [22] explored the application of functional size measurement at the hardware level and demonstrated how COSMIC Functional Size Measurement (FSM) can be utilized to measure the functionality of compiled assembly programs. Nonetheless, their investigation relies on an illustrative example and warrants further research for more definitive conclusions on whether the COSMIC FSM is applicable for low-level implementations of quantum software requirements. To date, there have been three studies concerning functional size measurement (FSM): [5], [19], [20]. All these studies utilized the COSMIC FSM methodology to measure the size of quantum software at three different abstraction levels: subsystem level, functional requirements level, and circuit level. However, the feasibility of applying these methods to measure the size of quantum software has not yet been thoroughly explored. Therefore, FSM in the context of quantum software requirements remains an ongoing research area.

## 6. Conclusion

In this study, we aimed to shed light on issues relevant to quantum software sizing to raise awareness among researchers. Their motivation was to open avenues for further exploration and discussion, encouraging the development of a deeper understanding of the complexities involved. In conclusion, the exploration of quantum software characteristics in the literature highlights its distinctive nature compared with classical software.

Quantum computation fundamentally diverges from classical approaches, necessitating specialized methodologies for requirement analysis and size measurement. Existing studies emphasize the need for tailored approaches to address quantum-specific functional and nonfunctional requirements, including considerations for error correction and system reliability. Furthermore, the definition and specification of functional requirements for quantum software demand reevaluation because of its lower abstraction level compared to its classical counterparts. Questions persist regarding how to measure the size of quantum software: Should it be defined functionally, or should the measurement criteria be expanded to encompass specific quantum software attributes?

Although studies utilizing the COSMIC FSM methodology offer insights into measuring the size of quantum software, further research is needed to ascertain its applicability at lower abstraction levels. Despite these advancements, the feasibility of applying existing methodologies to measure the size of quantum software remains an ongoing area of investigation.

## Acknowledgements

This work was supported by Mitacs through the Mitacs Elevate Program and COSMIC Group. We would also like to thank Prof. Alain Abran for his valuable contributions in reviewing this article.

## References

[1] S. Ali, T. Yue, and R. Abreu, “When software engineering meets quantum computing,” *Commun. ACM*, vol. 65, no. 4, pp. 84–88, 2022.

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

[3] Y. Zhang, H. Deng, Q. Li, H. Song, and L. Nie, “Optimizing quantum programs against decoherence: Delaying qubits into quantum superposition,” presented at the 2019 International Symposium on Theoretical Aspects of Software Engineering (TASE), IEEE, 2019, pp. 184–191.

[4] B. Weder, J. Barzen, F. Leymann, and D. Vietz, “Quantum software development lifecycle,” in *Quantum Software Engineering*, Springer, 2022, pp. 61–83.

[5] K. Khattab, H. Elsayed, and H. Soubra, “Functional Size Measurement Of Quantum Computers Software.,” presented at the IWSM-Mensura, 2022.

[6] N. Dey, M. Ghosh, and A. Chakrabarti, “QDLC--The Quantum Development Life Cycle,” *ArXiv Prepr. ArXiv201008053*, 2020.

[7] I.-D. Gheorghe-Pop, N. Tcholtchev, T. Ritter, and M. Hauswirth, “Quantum devops: Towards reliable and applicable nisq quantum computing,” presented at the 2020 IEEE Globecom Workshops (GC Wkshps, IEEE, 2020, pp. 1–6.

[8] K. Ahmad, M. Abdelrazek, C. Arora, M. Bano, and J. Grundy, “Requirements engineering for artificial intelligence systems: A systematic mapping study,” *Inf. Softw. Technol.*, vol. 158, p. 107176, 2023.

[9] T. Yue, S. Ali, and P. Arcaini, “Towards Quantum Software Requirements Engineering,” presented at the 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2023, pp. 161–164.

[10] L. Saraiva, E. H. Haeusler, V. G. Costa, and M. Kalinowski, “Non-Functional Requirements for Quantum Programs.,” presented at the Q-SET@ QCE, 2021, pp. 89–73.

[11] J. Zhao, “Quantum software engineering: Landscapes and horizons,” *ArXiv Prepr. ArXiv200707047*, 2020.

[12] P. E. Z. Junior and V. V. de Camargo, “A systematic mapping on quantum software development in the context of software engineering,” *ArXiv Prepr. ArXiv210600926*, 2021.

[13] Abran, A., Moore, J.W., Bourque, P. and Dupuis, R. (eds.), *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society Press, 2004.

[14] *ISO/IEC (2011). ISO/IEC 25010 – “Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE),”* 2011.

[15] C. A. Pérez-Delgado and H. G. Perez-Gonzalez, “Towards a quantum software modeling language,” presented at the Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, 2020, pp. 442–444.

[16] R. Pérez-Castillo, L. Jiménez-Navajas, and M. Piattini, “Modelling quantum circuits with UML,” presented at the 2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE), IEEE, 2021, pp. 7–12.

[17] M.-A. Sicilia, M. Mora-Cantallops, S. Sánchez-Alonso, and E. García-Barriocanal, “Quantum Software Measurement,” in *Quantum Software Engineering*, Springer, 2022, pp. 193–208.

[18] J. Zhao, “Some size and structure metrics for quantum software,” presented at the 2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE), IEEE, 2021, pp. 22–27.

[19] A. Lesterhuis, *COSMIC Measurement Manual for ISO 19761, Measurement of Quantum Software Circuit Strategy, a circuit-based Measurement Strategy*, 2024.

[20] F. Valdes-Souto, H. G. Perez-Gonzalez, and C. A. Perez-Delgado, “Q-COSMIC: Quantum Software Metrics Based on COSMIC (ISO/IEC19761),” *ArXiv Prepr. ArXiv240208505*, 2024.

[21] International Organization for Standardization, *ISO/IEC 19761: 2011, Software Engineering – COSMIC: A functional size measurement method,* Geneva., 2011.

[22] A. Darwish and H. Soubra, “COSMIC Functional Size of ARM Assembly Programs.,” presented at the IWSM-Mensura, 2020.