Quantum Software Sizing: Contemporary Interpretations and Approaches

Hassan SOUBRA[1]

[1]Ecole Centrale d'Electronique-ECE Lyon, 24 rue Salomon Reinach, 69007 Lyon

Abstract

Conventional software sizing approaches initially centered on metrics like lines of code, gradually transitioning to more refined measurements such as function points. However, these approaches could not be directly applicable to quantum software due to the fundamental differences between classical and quantum computing paradigms. In quantum software sizing, factors such as the number of qubits required, the depth of quantum circuits, the connectivity requirements of qubits, the error rates of quantum gates, and the complexity of the quantum algorithms play crucial roles. Additionally, considerations such as the choice of quantum programming language, quantum hardware platform, and optimization techniques also impact the overall size estimation. This paper provides an overview of quantum software sizing, highlighting initial exploration and classification of sizing measurement concepts of Quantum Software.

Keywords

Quantum Software, Quantum Software Sizing, Quantum Software Engineering, Metrics, Measurement

1. Introduction

Within the context of "classical" computers, Software sizing refers to the process of measuring the size of a software project, which is crucial for various aspects of software development such as project cost, effort, schedules, and duration [1] [2]. Accurate software sizing is essential for effective project management and decision-making, as it provides valuable information for software project development [3]. Different methods are employed for software sizing estimation, including lines of code (LOC) and function point analysis (FPA) [4]. The size estimation process becomes more accurate as the software model evolves and requirements become clearer [5]. Risks associated with software sizing, especially in dynamic and parallel processing environments, highlight the importance of precise estimation to avoid negative consequences and project suspension.

The evolution of quantum computers stems from integrating principles of Quantum Mechanics into computing, enabling the representation of multiple states simultaneously through Qubits, unlike classical computers that operate on binary logic [6]. Quantum Computing (QC) has made significant strides in recent years, offering exponential speed and scalability by leveraging quantum phenomena like superposition and entanglement [6]. Quantum Computers have shown immense potential in various fields, including Quantum Machine Learning (QML), where the fusion of quantum and machine learning algorithms has yielded exceptional results.

IWSM-MENSURA'24, September 30 - October 4, 2024, Montreal, Canada

hsoubra@ece.fr (H. SOUBRA)

The transition from classical to quantum computing has paved the way for solving complex and computationally intractable problems efficiently, marking a significant milestone in the realm of computing [7].

Within the realm of quantum computing, the distinction between software and hardware is nuanced compared to classical computing paradigms. Quantum hardware encompasses the physical components responsible for implementing quantum operations and storing quantum information, such as qubits, gates, and registers. Conversely, quantum software comprises abstract algorithms and instructions crafted to manipulate quantum states for specific computational or simulation tasks. While conceptually distinct, quantum software and hardware are intricately interconnected in practice. Quantum software development necessitates careful consideration of the capabilities and limitations of specific quantum hardware architectures, with close collaboration between quantum software developers and hardware engineers to optimize algorithms for the underlying hardware. Conversely, advancements in quantum hardware often drive progress in quantum software development, emphasizing the symbiotic relationship between the two domains. This interdependence underscores the unique challenges and opportunities in the development and deployment of quantum computing technologies.

In essence, the evolution of quantum computing has ushered in a paradigm shift by integrating principles of Quantum Mechanics into computing, enabling the representation of multiple states simultaneously through Qubits, unlike classical computers that operate on binary logic. Quantum Computing (QC) has witnessed remarkable progress, offering exponential speed and scalability by leveraging quantum phenomena like superposition and entanglement.

This paper explores the nuanced landscape of quantum software sizing, delving into its contemporary interpretations and approaches within the field of quantum computing. By examining key metrics, methodologies, and the intricate interplay between quantum hardware and software, this paper aims to provide insights that contribute to the ongoing dialogue surrounding quantum software development and optimization. Through a initial analysis of current practices and emerging trends, this paper seeks to elucidate the challenges, opportunities, and future directions in the realm of quantum software sizing.

The rest of this paper is organized as follows: Section 2 offers an overview of the key principles underpinning software sizing in the context of "classical" computing. In Section 3, we delve into the nuances of quantum software sizing, examining its importance and the various factors that influence it. Building upon this foundation, Section 4 provides current approaches to quantum software sizing, elucidating the approaches and metrics employed. Furthermore, Section 5 presents our conclusions drawn from the preceding discussions, summarizing key insights and highlighting avenues for future research in quantum software sizing.

2. Overview of the Key principles of software sizing

Accurate software sizing principles form the cornerstone of effective project planning and management, underlining the need for precise estimation of a software project's size [8] [1] [9] [10]. At the heart of these principles lie two primary metrics: Lines of Code (LOC) and Function Point Analysis (FPA). Historically, software sizing relied heavily on LOC metrics, which quantified the size of a software project based on the number of lines of code written. However,

the evolving landscape of software development has highlighted the limitations of LOC metrics, particularly in capturing the complexity and functionality of modern software systems. In contrast, Function Point Analysis (FPA) has emerged as a more relevant and comprehensive measurment for software sizing. FPA measures the functional size of a software application based on the complexity and functionality of its components, providing a more nuanced and accurate representation of software size. While both LOC and FPA have their strengths and weaknesses [11], the adoption of FPA is increasingly favored due to its ability to capture the intricacies of modern software systems more effectively.

Effective software sizing estimation is paramount for determining various aspects of software project development, including project costs, effort, schedules, and duration. The significance of accurate size estimation is particularly pronounced at the project's inception when high-level decisions are made based on initial requirements. However, size estimation poses inherent challenges and risks, requiring a principled approach rooted in sound theoretical foundations. Function points, in this regard, emerged as valuable tools for size estimation, offering benefits when applied across different stages of the software development lifecycle. By utilizing function points, organizations can derive cost-effective insights into software size, enabling informed decision-making and resource allocation. Emphasizing a systematic and principled approach to measuring software size ensures that project planning and management are conducted with precision and foresight, laying a robust foundation for successful software development endeavors.

3. Discussing Quantum software sizing

Discussing quantum software sizing is highly relevant in the field of quantum computing due to several key reasons. Firstly, just like in classical computing, measuring the performance of quantum software is crucial for optimization. Different measures and metrics should provide valuable insights into the efficiency of quantum algorithms and programs. Optimizing these metrics enables quantum software developers to enhance the speed and accuracy of quantum computations, thereby advancing the capabilities of quantum computing technology.

Secondly, effective resource management is essential in quantum computing, given the limited resources available, such as the number of qubits and the fidelity of quantum gates. Metrics and measurements aid in efficiently managing these resources by providing an understanding of the resource requirements of quantum algorithms. This understanding allows researchers to assess the feasibility of running algorithms on current or near-term quantum hardware, guiding resource allocation and optimization efforts.

Moreover, quantum hardware is still inherently prone to errors due to factors such as decoherence and gate imperfections. Metrics for error rates, fidelity, and error-correction capabilities are crucial for characterizing and mitigating errors in quantum computations. By measuring error rates and understanding error patterns, quantum software developers can design error-correction codes and fault-tolerant algorithms, improving the reliability and robustness of quantum computing systems.

Furthermore, metrics play a vital role in algorithmic development by facilitating the comparison of different quantum algorithms for the same task and are used to evaluate the effectiveness

of various quantum algorithms, driving innovation and optimization in algorithm design.

Additionally, the development of standardized metrics and benchmarks is essential for comparing the performance of different quantum software implementations and hardware platforms. Standardized metrics enable fair comparisons and facilitate the exchange of best practices among researchers and practitioners in the quantum computing community, fostering collaboration and progress in the field.

Finally, metrics could play a significant role in tracking progress in quantum computing, including the demonstration of quantum supremacy [12]. By defining clear metrics and measuring progress against them, researchers can track advancements in quantum computing and set goals for future developments, driving innovation and practical applications of quantum computing technology.

In the realm of quantum computing, the distinction between software and hardware can be somewhat blurred compared to classical computing:

Quantum Hardware: This refers to the physical devices that implement quantum operations and store quantum information. Quantum hardware includes components such as qubits (quantum bits), quantum gates, and quantum registers. Examples of quantum hardware include superconducting qubits, trapped ions, and topological qubits.

Quantum Software: Quantum software consists of algorithms, protocols, and programming languages designed to run on quantum hardware. This software defines the logic and operations that manipulate quantum states to perform specific computations or simulations. Quantum software may include programming languages like Qiskit, Cirq, or Quipper [13], as well as algorithms like Grover's Algorithm or Shor's Algorithm.

Now, to the question of discernibility:

- **At a conceptual level**: The concepts of quantum hardware and software are distinct. Hardware refers to the physical implementation of quantum systems, while software refers to the abstract algorithms and instructions that manipulate quantum states. Just like in classical computing, where hardware refers to the physical components and software refers to the programs that run on them, the same distinction applies in quantum computing.

- **At a practical level**: Quantum software often interacts closely with the capabilities and limitations of quantum hardware. Quantum algorithms and programs need to be optimized to run efficiently on specific quantum hardware architectures, taking into account factors such as qubit connectivity, error rates, and gate fidelities. Quantum software developers often work closely with quantum hardware engineers to ensure that algorithms are tailored to the capabilities of the underlying hardware.

- **In terms of development and deployment**: Quantum software development typically involves writing code in quantum programming languages, simulating algorithms on classical computers, and then running them on actual quantum hardware. Quantum hardware, on the other hand, undergoes physical fabrication and testing in specialized laboratories. While the processes for developing quantum software and hardware are distinct, they are often intertwined, with advancements in one area driving progress in the other.

Given the close interconnection between quantum software and hardware in practice, although they maintain conceptual distinction, quantum algorithms are specifically crafted to leverage the functionalities of particular quantum hardware architectures. Consequently, discussions on Quantum "Software" Sizing Approaches cannot presently be detached from their

intimate association with Quantum hardware.

4. Current Quantum "Software" Sizing Approaches

Various software sizing approaches in quantum computing include metrics for measuring the size and structure of quantum software [14], identifying qubits suitable for circuit resizing, and combining theoretical views with practical tasks through model-based simulations [15] [16]. These approaches aim to evaluate quantum software rigorously and quantitatively, and optimize circuit execution on limited qubit systems. By utilizing metrics at different abstraction levels, selecting qubits for serial execution, and employing model-based simulations with gradient-based optimization, these approaches enhance the performance and reliability of quantum circuits on small hardware, improving the execution of large circuits and minimizing errors.

4.1. Quantum Volume

Quantum volume [17] is a crucial metric in quantum computing that quantifies the number of usable qubits on a quantum computer. It serves as a comprehensive benchmark for assessing the performance of quantum computers, considering factors like hardware improvements, software enhancements, and error mitigation techniques. This metric has been extended to include various circuit shapes, known as Quantum Volumetric Classes [18] [19], to better align with specific quantum computing applications. The concept of effective quantum volume has been introduced, which incorporates error mitigation techniques like Digital Zero-Noise Extrapolation [20] to enhance the quantum volume of quantum processors, showcasing improvements over the vendor-measured quantum volume. Understanding quantum volume is essential for evaluating the capabilities and advancements of quantum computing systems.

While hardware enhancements directly boost quantum volume, software improvements, particularly in compilers, also play a significant role in increasing this metric. Notably, error mitigation techniques, a form of indirect compilation, have been shown to effectively enhance quantum volume without increasing the number of overall samples required . Quantum volume's ability to capture both hardware and software advancements makes it a valuable tool for evaluating quantum computer capabilities and the impact of error correction strategies on overall performance.

The key components of quantum volume calculation include factors like qubit number, fidelity, connectivity, and other crucial quantities for building functional quantum devices. The quantum volume test aims to provide a single-number metric for assessing a quantum computer's overall capability, although a complete understanding of its limitations and operational significance is still evolving. To enhance the test's efficacy, researchers have explored design aspects, error sensitivity, passing criteria, and implications of passing the test on a quantum computer's abilities . Additionally, efforts have been made to develop efficient algorithms for estimating heavy output probabilities under various error models and compiler optimization choices, predicting performance benchmarks for future quantum systems. This comprehensive approach to quantum volume calculation is essential for evaluating and advancing quantum computing technologies.

Noise significantly impacts the accuracy of quantum volume calculations by introducing errors that affect the reliability of quantum computations. Various types of noise, such as decoherence, gate errors, readout errors, leakage, and crosstalk, can degrade the quality of quantum circuits. To address these challenges, researchers have developed classical simulation algorithms like LOWESA [21], which estimate expectation values of noisy parameterized quantum circuits efficiently. Evaluating noise models systematically is crucial for understanding and predicting the impact of errors on quantum computations. By constructing accurate noise models and benchmarking them against hardware experiments, researchers aim to mitigate errors and enhance the accuracy of quantum computing applications.

4.2. Error Rates

The most common types of errors in quantum software include Program anomaly bugs, Configuration bugs, and Data type and structure bugs [22]. These bugs are often found in components like the compiler, gate operation, and state preparation components in quantum programming projects. To detect and measure these errors, it is crucial to conduct empirical studies on bug reports from quantum software projects, as done in the research. Additionally, identifying and categorizing bug patterns specific to quantum programming languages, such as Qiskit, can help in understanding and preventing common mistakes in quantum programs [23]. Furthermore, a detailed analysis of real-world bugs in quantum computing platforms can provide insights into quantum-specific bugs and recurrent bug patterns, aiding developers in avoiding errors and assisting tool builders in enhancing bug prevention and detection mechanisms.

Entanglement errors can significantly impact quantum software performance [24]. The entanglement degree in data can have a dual effect on prediction error, depending on the number of permitted measurements. Quantum error correction (QEC) codes play a vital role in mitigating detriments induced by noise, preserving entanglement, and enhancing the performance of quantum protocols even under larger noise amplitudes. Additionally, entanglement analysis in quantum programs is crucial for understanding quantum behavior and preventing entanglement-induced errors. Static entanglement analysis methods, like constructing interprocedural control flow graphs, help identify entanglement interactions within and between modules, ultimately improving the reliability and security of quantum programs. These findings underscore the critical role of managing entanglement errors for optimizing quantum software performance.

4.3. Connectivity

The connectivity of qubits within a quantum computer is essential for quantum software sizing. Better connectivity enables more efficient quantum operations and interactions, affecting the scalability and complexity of quantum software applications. Qubit connectivity significantly influences quantum software scalability. Constraints on qubit connectivity, such as those in superconducting qubits and quantum dots, can lead to challenges in implementing unrestricted quantum circuits efficiently [25]. However, innovative approaches like efficient qubit-mapping methods can mitigate these limitations by redesigning circuits to optimize connectivity, reducing additional gates and runtime while enhancing circuit stability [26].

Qubit connectivity plays a crucial role in quantum error correction schemes. Different error

correction codes, such as the surface code, XZZX code, reduced-connectivity surface code, XYZ matching code, and Floquet code [27], exhibit varying advantages in terms of error threshold, connectivity, and logical qubit encoding, depending on the qubit connectivity. For instance, the surface code relies on square-grid connectivity, which allows for high error thresholds and logical qubit storage. Quantum low-density parity-check (LDPC) codes are also impacted by qubit connectivity, with the graph separator of the connectivity graph influencing code limitations and the ability to construct good codes. Sparse connectivity in superconducting quantum computers leads to experimental overheads, which can be mitigated by techniques like virtual two-qubit gates for error suppression. Therefore, qubit connectivity directly influences the effectiveness and efficiency of quantum error correction strategies.

4.4. Functional Size Measurement

Functional Size Measurement tailored for quantum software. It encompasses data movements through the introduction of novel data-movement types and quantifies quantum data movements using COSMIC Functional Points: Q-COSMIC, a technique for measuring the functional size of quantum software [28] and a functional size measurement (FSM) procedure for Quantum Computer Software with functional requirements implemented in Qiskit [29]. Both approaches are based on the COSMIC-ISO 19761 delineate the functional user, classical, and quantum parts of quantum software as separate entities with boundaries, providing a comprehensive measurement approach.

4.5. Basic Quantum Size Metrics

Code Size, Design Size, and Specification Size metrics. Examples of Code Size metrics include Lines of Code (LOC), while Design Size metrics utilize quantum architectural description language (qADL) and Quantum Unified Modeling Language (Q-UML) to specify architectures. Basic Structure metrics such as McCabe's Complexity Metric and Henry and Kafura's information flow Metric were also considered [14].

4.6. Quantum Circuit Metrics

Quantum circuit metrics are crucial for assessing various aspects of quantum computing. These metrics help in evaluating the performance, understandability, and applicability of quantum circuits [30]. One approach involves designing quantum circuits metrics for Bayesian optimization with Gaussian processes, which includes a new quantum gates distance to characterize gates' actions over quantum states[31]. Additionally, the concept of Quantum Volumetric Classes has been introduced to generalize the quantum volume metric, considering different circuit shapes and their scalability with problem sizes [18]. Furthermore, there is a focus on developing metrics to determine the successful execution of quantum circuits on gate-based quantum computers, emphasizing the importance of assessing current quantum computing capabilities [32]. Quantum circuit metrics play a crucial role in determining the efficiency of quantum algorithms. By leveraging families of quantum states that can be prepared with linear-size and depth circuits, quantum algorithms can achieve high precision initialization, enabling depth-efficient executions. Metrics-aware quantum algorithms, such as those utilizing

the quantum Fisher information matrix, optimize the distribution of samples between matrix and vector entries, leading to efficient estimation with fewer circuit repetitions. Techniques like gate re-ordering in the Quantum Approximate Optimization Algorithm (QAOA) reduce gate-count and circuit-depth, enhancing noise resilience and execution time. Additionally, novel approaches like the Projected-Variational Quantum Dynamics (p-VQD) algorithm offer efficient global optimization of variational parameters, significantly improving scalability for large parameterized quantum circuits.

4.7. Quality of hybrid code

Evaluating the quality of hybrid code [33], specifically targeting maintainability, is crucial as the ability to evolve is one of the pivotal characteristics in the still-developing quantum software industry. Despite the well-defined nature of quantum algorithms presently in use, the landscape of quantum technology is rapidly evolving, indicating ongoing advancements in the coming years. Consequently, quantum software is poised for continuous evolution, adapting to incorporate the latest innovations and enhancements in quantum technology.

Moreover, the current quantum market is fiercely competitive, emphasizing the significance of developing dependable and efficient quantum software. The ability to do so can dictate the success or failure of endeavors in this domain. Thus, [33] aims to introduce the inaugural maintainability metrics for quantum code, along with a framework for their assessment within hybrid software environments. Furthermore, it illustrates the practical application of these metrics by analyzing a prominent quantum algorithm, such as Shor's algorithm.

4.8. Quantum Processing Units- QPU Software related Metrics

Key performance metrics for quantum processing units (QPUs) include the Q-score Max-Clique metric for comparing different quantum computing paradigms [34], circuit depth, and gate count for solving NP-complete problems with tailored hardware properties [35]. Additionally, metrics like randomized benchmarking and quantum volume are crucial for assessing quantum platforms, with a recent focus on adapting these metrics for measurement-based quantum computing (MBQC) processors like photonic devices [36]. These metrics help in evaluating the speed, efficiency, and effectiveness of QPUs in performing quantum computations, guiding the development and optimization of quantum hardware for practical deployment and scalability.

Table 1 summarizes the main Quantum Software Sizing Measurable concepts.

5. Conclusion

In this paper, we have examined the contemporary interpretations and approaches to quantum software sizing within the rapidly evolving field of quantum computing. Through our exploration, several key insights have emerged.

Firstly, we have highlighted the importance of rigorous and quantitative evaluation of quantum software, considering factors such as quantum volume, qubit connectivity, and error rates. These metrics and measures provide valuable insights into the efficiency, reliability, and scala-

Table 1
Summary of Main Quantum Software Sizing Measurable Concepts

Measurable concepts	**Description**
Quantum Volume	Quantifies the number of usable qubits on a quantum computer.
Effective Quantum Volume	Incorporates error mitigation techniques to enhance quantum volume.
Qubit Connectivity	Influences the efficiency and scalability of quantum software applications.
Functional Size Measurement	Measures the functional size of quantum software, incorporating quantum data-movements.
Basic Quantum Size Metrics	Includes metrics such as Code Size, Design Size, and Specification Size.
Quantum Circuit Metrics	Assess various aspects of quantum circuits, including performance and understandability.
Quality of Hybrid Code	Evaluates the quality of hybrid quantum-classical code, focusing on maintainability.
Quantum Processing Units Metrics	Key performance metrics for quantum processing units (QPUs), such as Q-score Max-Clique and circuit depth.
Error Rates	Measures types of errors in quantum software, including Program anomaly bugs, Configuration bugs, and Data type and structure bugs.
Entanglement Errors	Impacts quantum software performance, requiring management for reliability and security.

bility of quantum computing systems, guiding optimization efforts and resource management strategies.

Secondly, our analysis has underscored the critical role of both hardware and software advancements in enhancing quantum computing capabilities. From improvements in quantum volume to the development of error mitigation techniques and hybrid code quality evaluation, advancements in both domains contribute synergistically to the overall performance of quantum systems.

Furthermore, our examination of various sizing approaches has revealed the interdisciplinary nature of quantum software development, requiring collaboration between quantum software developers, quantum hardware engineers, and researchers across multiple domains. By leveraging novel data-movement types, advanced circuit metrics, and hybrid code evaluation techniques, the quantum computing community can drive innovation and progress in quantum software sizing.

As we look to the future, it is evident that quantum software sizing will continue to be a dynamic and evolving area of research. With ongoing advancements in quantum hardware, software, and algorithms, the landscape of quantum computing will continue to evolve, presenting new challenges and opportunities for quantum software developers.

References

[1] A. Sathesh, Y. B. Hamdan, Analysis of software sizing and project estimation prediction by machine learning classification, Journal of Ubiquitous Computing and Communication Technologies 3 (2021) 303–313.

[2] A. L. AL-Saleem, Y. H. Asma'a, Software size estimation: A survey (2022).
[3] J. T. Mesia Dhas, et al., The functional and storage risks associated to the size estimation of parallel computing applications, in: Advances in Parallel Computing Algorithms, Tools and Paradigms, IOS Press, 2022, pp. 373–379.
[4] M. Nasir, A survey of software estimation techniques and project planning practices, in: Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06), IEEE, 2006, pp. 305–310.
[5] A. Abran, C. Symons, C. Ebert, F. Vogelezang, H. Soubra, Measurement of software size: Contributions of cosmic to estimation improvements, in: The International Training Symposium, At Marriott Bristol, United Kingdom, 2016, pp. 259–267.
[6] P. B. Upama, M. J. H. Faruk, M. Nazim, M. Masum, H. Shahriar, G. Uddin, S. Barzanjeh, S. I. Ahamed, A. Rahman, Evolution of quantum computing: A systematic survey on the use of quantum computing tools, in: 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, 2022, pp. 520–529.
[7] R. Pandey, P. Maurya, G. D. Singh, M. S. Faiyaz, Evolutionary analysis: Classical bits to quantum qubits, in: Quantum Computing: A Shift from Bits to Qubits, Springer, 2023, pp. 115–129.
[8] E. G. Ng'ang'a, A survey on software sizing for project estimation (2015).
[9] P. M. Khan, M. Beg, Application of sizing estimation techniques for business critical software project management, arXiv preprint arXiv:1405.6335 (2014).
[10] F. G. Wilkie, I. R. McChesney, P. Morrow, C. Tuxworth, N. Lester, The value of software sizing, Information and Software Technology 53 (2011) 1236–1249.
[11] L. O. Ejiogu, Attribute-based model of software size, in: Software Measurement: Current Trends in Research and Practice, Springer, 1999, pp. 81–92.
[12] I. Olga, K. Vladimir, T. Olga, U. Sergey, F. Toshio, Quantum supremacy in end-to-end intelligent it. pt. i: Quantum software engineering-quantum gate level applied models simulators, Systems Analysis in Science and Education 1 (2020) 52–84.
[13] A. Bai, Report on formal verification in quantum programming (2023).
[14] J. Zhao, Some size and structure metrics for quantum software, in: 2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE), IEEE, 2021, pp. 22–27.
[15] M. Sadeghi, S. Khadirsharbiyani, M. T. Kandemir, Quantum circuit resizing, arXiv preprint arXiv:2301.00720 (2022).
[16] S. Niu, A. Hashim, C. Iancu, W. A. de Jong, E. Younis, Powerful quantum circuit resizing with resource efficient synthesis, arXiv preprint arXiv:2311.13107 (2023).
[17] K. Kechedzhi, S. Isakov, S. Mandrà, B. Villalonga, X. Mi, S. Boixo, V. Smelyanskiy, Effective quantum volume, fidelity and computational cost of noisy quantum processing experiments, Future Generation Computer Systems 153 (2024) 431–441.
[18] K. Miller, C. Broomfield, A. Cox, J. Kinast, B. Rodenburg, An improved volumetric metric for quantum computers via more representative quantum circuit shapes, arXiv preprint arXiv:2207.02315 (2022).
[19] O. Ghoniem, H. Elsayed, H. Soubra, Quantum gate count analysis, in: 2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS), IEEE, 2023, pp. 190–197.
[20] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, W. J. Zeng, Digital zero noise extrapolation

for quantum error mitigation, in: 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2020, pp. 306–316.
[21] M. S. Rudolph, E. Fontana, Z. Holmes, L. Cincio, Classical surrogate simulation of quantum systems with lowesa, arXiv preprint arXiv:2308.09109 (2023).
[22] H. Li, F. Khomh, L. Tidjon, et al., Bug characteristics in quantum software ecosystem, arXiv preprint arXiv:2204.11965 (2022).
[23] M. Paltenghi, M. Pradel, Bugs in quantum computing platforms: an empirical study, Proceedings of the ACM on Programming Languages 6 (2022) 1–27.
[24] T. Shang, J. Liu, Secure quantum network coding theory, Springer, 2020.
[25] R. Li, L. Petit, D. P. Franke, J. P. Dehollain, J. Helsen, M. Steudtner, N. K. Thomas, Z. R. Yoscovits, K. J. Singh, S. Wehner, et al., A crossbar network for silicon quantum dot qubits, Science advances 4 (2018) eaar3960.
[26] C.-Y. Huang, W.-K. Mak, Efficient qubit routing using a dynamically-extract-and-route framework, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2024).
[27] B. Hetényi, J. R. Wootton, Tailoring quantum error correction to spin qubits, Physical Review A 109 (2024) 032433.
[28] F. Valdes-Souto, H. G. Perez-Gonzalez, C. A. Perez-Delgado, Q-cosmic: Quantum software metrics based on cosmic (iso/iec19761), arXiv preprint arXiv:2402.08505 (2024).
[29] K. Khattab, H. Elsayed, H. Soubra, Functional size measurement of quantum computers software, Proceedings of the 31st IWSM-Mensura, Izmir, Turkey (2022).
[30] J. A. Cruz-Lemus, L. A. Marcelo, M. Piattini, Towards a set of metrics for quantum circuits understandability, in: International Conference on the Quality of Information and Communications Technology, Springer, 2021, pp. 239–249.
[31] T. Duong, S. T. Truong, M. Pham, B. Bach, J.-K. Rhee, Quantum neural architecture search with quantum circuits metric and bayesian optimization, in: ICML 2022 2nd AI for Science Workshop, 2022.
[32] M. Salm, J. Barzen, F. Leymann, B. Weder, About a criterion of successfully executing a circuit in the nisq era: what wd 1/ eff really means, in: Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software, 2020, pp. 10–13.
[33] A. Díaz Muñoz, M. Rodríguez Monje, M. G. Piattini Velthuis, Towards a set of metrics for hybrid (quantum/classical) systems maintainability., JUCS: Journal of Universal Computer Science 30 (2024).
[34] W. van der Schoot, R. Wezeman, N. M. Neumann, F. Phillipson, R. Kooij, Q-score max-clique: The first quantum metric evaluation on multiple computational paradigms, arXiv preprint arXiv:2302.00639 (2023).
[35] H. Safi, K. Wintersperger, W. Mauerer, Influence of hw-sw-co-design on quantum computing scalability, in: 2023 IEEE International Conference on Quantum Software (QSW), IEEE, 2023, pp. 104–115.
[36] Y. Zhang, D. Niu, A. Shabani, H. Shapourian, Quantum volume for photonic quantum processors, Physical Review Letters 130 (2023) 110602.