












Improving the Quality of Quantum Services Generation Process: Controlling Errors and Noise

Jaime Alvarado-Valiente¹  , Javier Romero-Álvarez¹ , Danel Arias² ,
Erik B. Terres² , Jose Garcia-Alonso¹ , Enrique Moguel³ ,
Pablo García Bringas² , and Juan M. Murillo³ 

¹ Escuela Politécnica, Quercus Software Engineering Group,
University of Extremadura, Cáceres, Spain
{jaimeav,jromero,jgaralo}@unex.es

² University of Deusto, Bilbao, Spain

danel.arias@opendeusto.es, {e.terres,pablo.garcia.bringas}@deusto.es

³ CénitS-COMPUTAEX, Cáceres, Spain
enrique@unex.es, director@cenits.es

Abstract. As the industry moves towards practical applications of quantum computing, it faces significant obstacles such as specific platform dependency and lack of mature tools. These obstacles make the creation of quantum applications a slow and complex process that requires specialized knowledge of quantum mechanics and computer science, which compromises the quality of quantum services. Therefore, the need to ensure an adequate level of quality in quantum software is fundamental. To address these challenges, this work proposes a process that enables developers to create high-quality quantum services in an automated and standardized way, using an extension of the OpenAPI specification. Furthermore, we analyze the challenges faced by NISQ devices, the most advanced quantum computers available today, due to errors and noise such as decoherence, gate errors, and readout errors. This process will make it possible to measure, at runtime, the stability and fidelity of the quantum circuits included in the generated quantum services.

Keywords: Quantum Computing · Quantum Services · Quantum Program Security · Quantum Code Analysis

1 Introduction

Quantum computing is a computational model that leverages the principles of quantum mechanics to manipulate and process information. This computing paradigm holds the potential to deliver high computational capacity, thereby allowing for the resolution of problems that have previously eluded classical computing, such as those that fall within the complexity class of BQP [1]. Therefore, the advent of quantum computing has garnered the attention of prominent technology companies like Amazon, IBM, or Google have invested extensively in

developing new quantum machines and providing them to users through their cloud platforms [2].

The utilization of quantum computers via cloud platforms, which are offered by different providers, has similarities with classical computing and service-oriented architectures. This entails that quantum computing may be employed within classical-quantum hybrid architectures, wherein both technologies contribute their resources in the form of services [3]. To create high-quality quantum services and hybrid architectures, developers require appropriate tools and techniques that enable them to attain the desired security standards [4]. It should be noted that the absence of adequate software engineering methods for quantum services presents various challenges, including the low abstraction level that developers must work with and the absence of integration, deployment, or quality and security control mechanisms for the software they develop [5].

Consequently, various solutions are emerging to bridge the divide between classical processes and quantum computing and tackle these problems [6]. However, quantum devices today face significant challenges due to the presence of errors and noise, which limits their scalability and usability. Therefore, controlling and improving the quality of new quantum systems is essential [7]. For example, Noisy Intermediate-Scale Quantum (NISQ) devices, which are the most advanced quantum computers available today [8], suffer from various types of noise and errors, such as decoherence, gate errors, and readout errors, among others. Thus, understanding the effects of noise and errors in NISQ devices is crucial for quantum software development [8]. When developing a quantum circuit, the results obtained in a NISQ backend may be misleading due to errors and noise. Therefore, it is crucial to evaluate the circuit's stability. Knowing how much the circuit deviates from the "perfect result" provides a direct measurement of its stability. As a result, two implementations with the same objective output but with a different sequence of operations may have different stability measurements.

Measuring the fidelity of a circuit involves comparing the distribution of results obtained in a real backend versus a simulated "perfect backend" (without noise or errors) after multiple executions of the circuit implementation. However, performing multiple executions is time and resource-intensive, as it requires repeated calls to services that offer real backends (which often have waiting queues, and can take minutes or even hours to execute). Additionally, executing in a simulator is generally slower as it must simulate the quantum characteristics. Thus, it is essential to know the stability of quantum circuits at runtime when developing a quantum program.

To this end, this paper proposes a tool for developers that combines an extension of the OpenAPI specification¹, along with a real-time measurement process of the quantum circuits to be executed. The OpenAPI extensions allow developers to define and generate quantum services in a similar and standardized way as classical services are defined. While the measurement process allows to

¹ <https://www.openapis.org/>.

check of the stability and evaluate the circuits included in the generated services at runtime.

To explain this, the organization of the paper is as follows: Sect. 2 provides an analysis of the background of the presented work and a discussion of the most relevant related work; Sect. 3 presents the proposal for the automatic generation and deployment of quantum services and the whole process of estimating and measuring the characteristics of the quantum circuits contained in those services; Sect. 4 includes the results of the measurements of the process; and finally Sect. 5 outlines some conclusions and presents the next steps to be carried out in the context of this research.

2 Background

NISQ computers represent a cutting-edge technology in the field of quantum computing, but their potential can be limited by environmental noise that can corrupt the qubit state. Such noise-induced decoherence is a fundamental issue that arises from the unavoidable interaction between qubits and their environment, which causes unpredictable and irreversible changes in the qubit state, leading to errors in measurement results. These errors pose challenges for addressing the quality and security requirements, which are crucial factors for the development of practical quantum applications [9].

In addition to decoherence, NISQ computers are also susceptible to operational errors, such as readout and gate errors. Readout errors arise when the measurement time exceeds the decoherence time of the qubits [10]. Gate errors, on the other hand, can result from various factors, including temperature fluctuations, electromagnetic interference, and environmental conditions [11]. Due to the limited number of qubits and short coherence times, the impact of gate errors can be significant and may impair the reliable execution of quantum algorithms.

To prevent malicious exploitation, quantum circuits must be secured against quantum errors that could be used by malicious agents to manipulate the correct state of a quantum circuit. It is therefore imperative for programmers to prioritize the development of quantum software with robustness that meets strict quality criteria. Any compromise in the security of quantum software can have severe consequences, particularly for sensitive applications such as cryptography or financial transactions, highlighting the criticality of ensuring error-free execution [12]. As a result, addressing quality and security requirements presents a significant challenge.

Therefore, the use of support tools during software design has proven to be an effective method for developers to improve the quality of their software. Despite this, there is a notable lack of such tools for quantum computing software, which presents a challenge for developers in this field. While support tools have been successful in enhancing the quality of classical computing software, their absence in the field of quantum computing adds an additional obstacle for developers tackling this complex issue [7].

Quantum simulations are currently the primary tool for approximating output noise in quantum circuits, but they have the main problem of a computationally intensive task. Obtaining the state vector of the output requires exponential memory with respect to the number of qubits, which can compromise the time of the simulation [13]. While certain circuits can be simulated in polynomial time, these are subject to specific constraints that do not apply to general-purpose analysis. For instance, stabilizer circuits can be efficiently simulated in classical systems, but they can only be written with CNOT, Hadamard, or Phase Gates [14]. This limitation presents a challenge in simulating general-purpose quantum circuits, which are the ones that will provide a quantum advantage over classical computers. Due to the exponentially increasing complexity of quantum circuits, it is not feasible to simulate larger circuits on classical systems. Thus, to estimate the error in quantum circuits in real-time, complete classical simulations are not a feasible solution.

In order to approximate quantum error, Aseguinolaza et al. [15] developed a highly accurate tool to approximate quantum error. Their algorithm utilizes a multiplicative noise rule to estimate noise, based on publicly available IBM computers noise models. They applied the algorithm to verify that the circuit fidelity aligns with the estimated error rate for various simple circuits, including the one-dimensional Ising model, quantum phase estimation, and the Grover algorithm.

The idea that simple mathematical algorithms make precise error rate estimations possible opens the door for more sophisticated error estimation algorithms for a broad range of general-purpose circuits. The development of these algorithms is essential to ensure that quantum software guarantees a minimum standard of quality and security. By limiting the execution of unsafe circuits, which have not passed the quality criteria, and by developing reliable and efficient error correction and error estimation methods, the quantum computing industry can mitigate the risks of quantum noise and ensure that the potential of this technology can be fully realized.

To ensure that quantum software meets strict quality and security requirements, a continuous integration and deployment process is essential. This process involves the use of OpenAPI for the generation of quantum services, facilitating their integration and use on different platforms and systems. By adapting these tools to work with quantum software [16], they can be combined with circuit measurements for the generation of quality services, which can help to detect potential errors and security issues in the code at an early stage.

3 Quantum Services Generation, Deployment and Error Measurement

The objective of this section is to provide an overview of the entire process of generating and deploying quantum services, along with a discussion of how to measure the quantum circuits that are part of these services.

3.1 OpenAPI Specification for Quantum Service Generation and Deployment

To achieve the continuous generation and deployment of quantum services, the OpenAPI specification has been utilized along with its code generator, which has been integrated into a workflow executed in the GitHub Actions tool².

The OpenAPI specification, also known as Swagger, is an industry-standard format used to describe RESTful APIs. It provides a machine-readable interface that enables automated documentation, client code generation, and testing of APIs. This specification, with the extension we have designed on it, enables the design of services including quantum properties.

With the new variables added, the developer can indicate which quantum circuit we want to encapsulate in the service—specifying the URL where it is available—, the quantum machine provider for which we want to generate it, the machine on which it will run, and the number of shots to launch—the latter two at runtime.

Once the developer has defined the services in the YAML specification, it goes through the OpenAPI code generator, which has also been modified to work with the new variables added to the specification. This generates circuit code ready for deployment and consumption. Once the code is ready, it is automatically deployed in a Docker container, and the user is given the URL where the API endpoints are available. This process is carried out once the developer makes a commit of the YAML in the GitHub repository, at which point the workflow execution is automatically triggered. The whole workflow can be seen in Fig. 1

In addition, as explained below, an enhancement has been added to analyze the circuits included in the specification.

3.2 Controlling Errors and Noise Through Estimation and Measurement

In order to evaluate the quality of the circuits included in the services and to find possible problems and vulnerabilities, a parallel process has been included to the deployment part that is in charge of predicting and estimating these possible errors in real execution time, as shown in the steps 6 and 7 of the workflow described above in Fig. 1. This error estimation acts as a monitoring tool for the developer that serves as a crucial component in understanding and mitigating the effects of noise and errors in quantum devices. For this purpose, within the workflow, the files containing only the code corresponding to the circuits are generated. These files are stored in a different repository to be analyzed and to obtain a report with these error measurements.

Therefore, in our process to estimate errors, we first studied the causes of errors through backend calibration analysis. We then measured the error of various circuits in different real backends. With the information collected, we developed a naïve estimation technique. Additionally, we further improved the error

² <https://github.com/features/actions>.

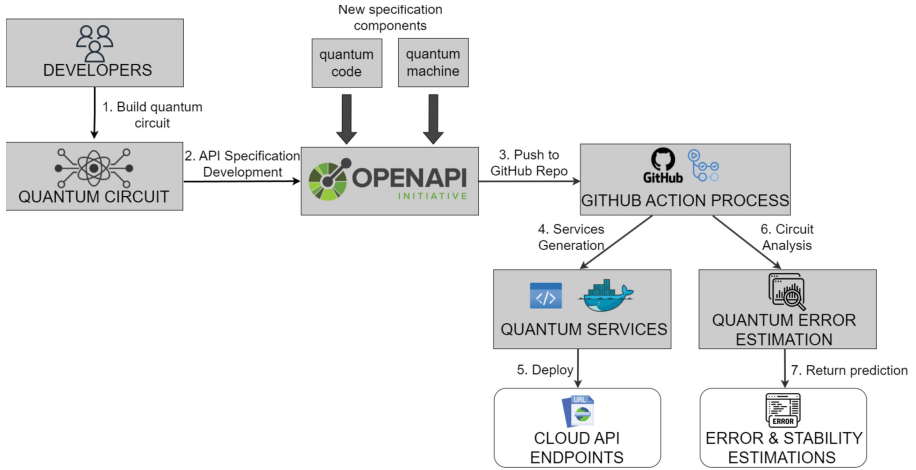


Fig. 1. Workflow for the generation and deployment of quantum services

estimation accuracy with machine learning approaches. In this section, we will dissect those steps with a further explanation.

We performed data analysis of all IBM Quantum backends, focusing on the error and stability of individual gates and qubits through the calibrations given by IBM Quantum. To investigate the error and stability of a quantum backend, we analyzed the calibration data provided by the backend. Our analysis includes examining error distributions, identifying the most impactful errors, and investigating correlations between the backends and other parameters.

Users can access this calibration data to gain insight into the behavior of the quantum backends. However, understanding the underlying complexities and their impact on the quantum circuit may require a deep level of knowledge which makes it challenging.

This calibration data includes various parameters that can help explain the noise and error susceptibility of the backend. The IBMQ backend calibration data includes parameters such as error probabilities and lengths of basic gates³, readout errors, probabilities of measuring 1 and 0 in certain states⁴, T1 and T2 relaxation times, and qubit frequency and anharmonicity. These parameters can help explain the noise and error susceptibility of the backend.

For instance, the error probability and length in nanoseconds of each basic gate can provide insights into how each gate contributes to the overall error of the backend. Similarly, the probability of error and length in nanoseconds of a readout can reveal the error in the measurement of the qubits. By analyzing T1 and T2 relaxation times in microseconds, the frequency of the qubits in GHz,

³ The basic gates create the basis on which the rest of the gates that make up the circuit are composed.

⁴ This consists on preparing a 0 state and measuring if a 1 is read and viceversa. Thus, these probabilities are related to the readout error.

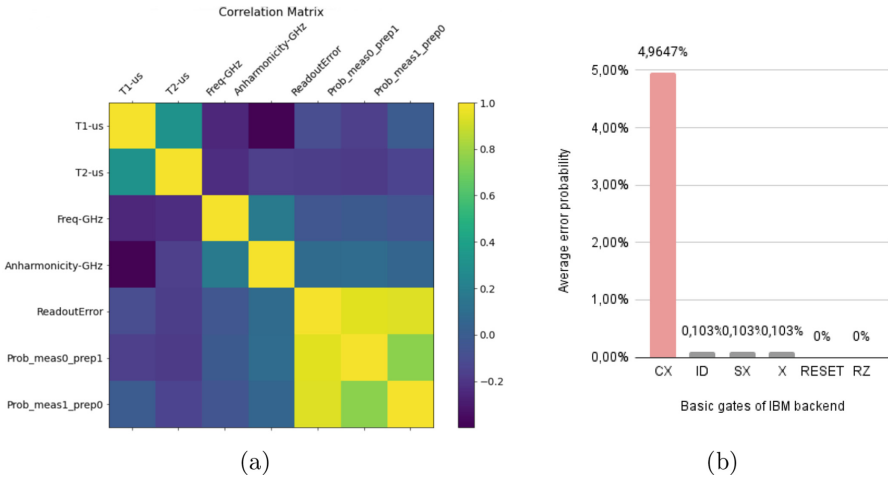


Fig. 2. (a) Correlation matrix of the different calibration metrics. The readout error is a combination of the probability of measuring 0 having prepared a 1 state and vice versa. (b) Average probability of error for the basic gates of IBM Quantum backends, based on the calibrations retrieved from IBM.

and the anharmonicity of the qubits in GHz can provide information about the coherence and stability of the qubits.

Based on our data analysis, we can draw several conclusions about the error and stability of the IBM Quantum backends:

1. Firstly, we found that there is no significant correlation between readout errors and gate errors. This means that these errors are independent of each other and improving one type of error may not necessarily lead to an improvement in the other. Therefore, these errors can be studied independently. The same happens with noise and stability calibrations, there is no significant correlation between these (see Fig. 2a).
2. Secondly, we identified CNOT (Controlled NOT) gates as the most error-prone gate across all backends. In fact, we found that there is a significant difference between the error rates of CNOT gates and the error rates of other gates (see Fig. 2b). This is an important finding as CNOT gates are widely used in quantum circuits, and improving their error rate could significantly improve the overall performance of the backends.
3. Finally, we found that while errors are backend-dependent, gate and readout errors are similar across all backends. Specifically, we found that the error rates are proportional when comparing different backends. This means that there is not a significant variation in error rates across backends, which is useful information for users to take into account when selecting a backend for their applications.

To estimate the error of a quantum circuit, we first developed a basic approach, which we refer to as the “naïve approach to quantum error estimation”.

This approach consists of obtaining the error probabilities of the basic gates and readout error of the backend b in which the circuit c will be executed. Next, we determine the number of each basic gate g_i and the number of measurements m that comprise c . Finally, we compute the total error probability as the sum of the product of the number of each basic gate and its corresponding error probability, plus the product of the number of measurements and the measurement error probability, divided by the total number of operations in the circuit:

$$\frac{1}{N_{ops}} \left(\sum_i^{n_g} g_i \cdot err(g_i) + m \cdot err(m) \right) \tag{1}$$

where N_{ops} is the total number of operations in c , n_g is the total number of basic gates in c , g_i is the number of the i -th basic gate in c , $err(g_i)$ is the error probability of the i -th basic gate in b , m is the number of measurements in c , and $err(m)$ is the measurement error probability in b .

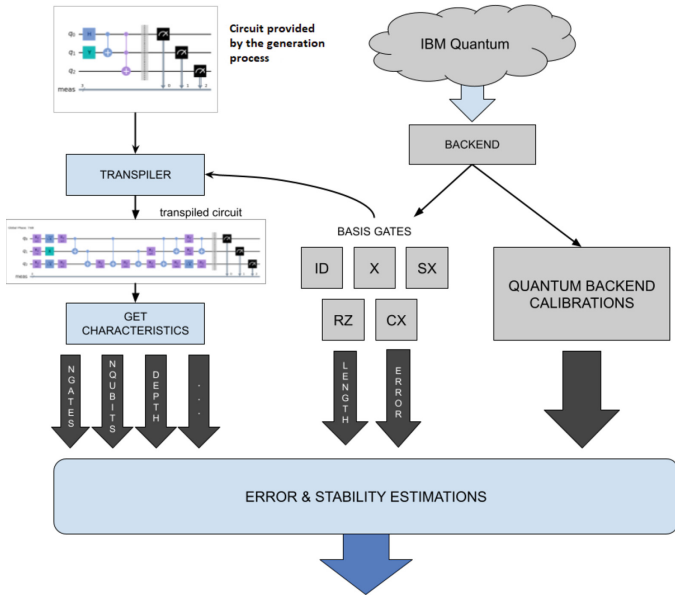


Fig. 3. Workflow of the quantum error estimation process for IBM Quantum backends

While the “naïve approach to quantum error estimation” is a simple and effective way to estimate the error of a quantum circuit, we also explored other alternatives. Such an approach involves using machine learning and AI techniques to model the error behavior of quantum devices. By analyzing large datasets of calibration data and experiment results, these models can learn to predict the error of a given circuit with higher accuracy than the naïve approach. To train

such models, we first created a dataset by running approximately 140000 experiments, using various circuits of different sizes and complexities. We then used this dataset to train two different models: an XGBoost Regressor and a Neural Network. The accuracy of these estimators is discussed in Sect. 4.

Therefore, we define the process for our quantum error estimation for a circuit and a backend as the following as shown in Fig. 3. First, we retrieve the backend information from the IBM Quantum cloud service. This information includes the backend calibration data and the gate basis. We then transpile the circuit to the basic gates and extract the characteristics (number of gates of each type, depth, number of qubits, number of classical bits, etc.). After that, we process all the data and we pass it to the estimator.

4 Evaluation

As we introduced before, we explored three different approaches to quantum error estimation: two machine learning-based methods using an XGBoost Regressor, a naïve estimation method, and a Neural Network.

Both these models were trained to predict the total error probability of a circuit given the circuit characteristics and the backend calibration data. We evaluated the performance of these models using cross-validation and compared them to the naïve approach. As it can be seen in Table 1, the XGBoost Regressor achieved significantly higher accuracy. In contrast, the naïve approach had nearly double the error and the Neural Network came last. These results demonstrate that the XGBoost Regressor model can achieve higher accuracy than the naïve approach in predicting the error of a given circuit.

Table 1. Mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) of the XGBoost Regressor, Naïve estimation, and Neural Network for quantum error estimation. The models were evaluated using 10-fold cross-validation on a dataset of approximately 55,000 experiments. The XGBoost Regressor achieved the highest accuracy among the three methods, with significantly lower MAE, MSE, and RMSE than the Naïve estimation and the Neural Network.

	MAE	MSE	RMSE
XGBoost Regressor	~0.0308	~0.0020	~0.0442
Naïve Estimation	~0.0629	~0.0120	~0.1096
Neural Network	~0.0868	~0.0112	~0.1052

To further evaluate the performance of the three error estimation methods, we executed the experiment on a set of specific circuits. For each circuit, we first used the estimators to predict the error probability and then ran the circuit on both a real quantum backend and a no-noise simulator. By comparing the results from the two runs, we calculated the actual error of the circuit. We

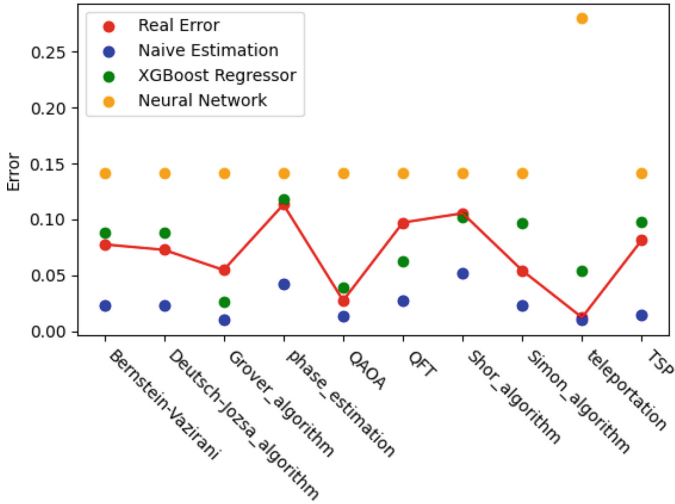


Fig. 4. Errors measured (in red) and errors estimated by the 3 different estimators in some popular and widely used quantum circuits. (Color figure online)

then compared the estimated error probability from each estimator to the actual error probability. This evaluation allowed us to assess the accuracy of the estimators in predicting the error probability of specific circuits. The results, shown in Fig. 4, demonstrate that the XGBoost Regressor model is the most accurate estimator, while the naïve method underestimates the error and the Neural Network performs unsatisfactorily. However, the tabular nature of the dataset may have contributed to the poor performance of the Neural Network, indicating the need for a more sophisticated preprocessing of the dataset. The underestimation of error by the naïve method, on the other hand, is explained by the fact that it does not take into account other error-inciting factors such as decoherence, relaxation or other forms of noise.

5 Conclusions

Quantum computing is a promising computational paradigm that offers solutions to previously unsolvable problems. However, the current state of quantum computing raises concerns regarding security and quality due to existing methods of quantum software usage and the lack of abstraction. Prioritizing research and development in quantum computing is essential to address these challenges. This paper proposes a solution that consists of adopting classical service engineering techniques and methods to address challenges in the development and utilization of quantum services.

To achieve this, a standardized method has been proposed for defining quantum services, which utilizes the OpenAPI specification. The process involves

generating source code for the quantum services using the OpenAPI Code Generator from the quantum circuit and OpenAPI specification. Additionally, to streamline the deployment of these services, a workflow has been created for deploying them in Docker containers using the GitHub Actions tool. This will help in managing and maintaining the services in a more organized way. To further enhance this process, a monitoring tool has been incorporated. This tool allows developers to assess the error estimates and vulnerabilities of the quantum circuits included in the services. With this tool, developers can easily detect and resolve any issues that may arise, thus ensuring that the quantum services are performing optimally.

Furthermore, we have demonstrated the importance of accurate quantum error estimation in the context of quantum services. Accurate quantum error estimation is crucial for achieving optimal performance and reducing the impact of noise on quantum circuits. Although quantum simulations are the primary tool for approximating output noise, complete classical simulations are not feasible for estimating errors in quantum circuits in real-time. The proposed monitoring tool can help developers easily detect and resolve errors, ensuring reliability and usefulness in practical applications. To achieve reliability and usefulness, accurate error estimation is necessary. While there is still much work to be done in mitigating the effects of noise and errors in quantum circuits, ongoing research in improving the quality of individual gates and measurements and creating standards for circuit design can help fully realize the potential of quantum computing in practical applications.

Acknowledgments. The authors would like to acknowledge the partial financial support by the Ministry of Science (QSERV project, PID2021-1240454OB-C31 and PID2021-124054OB-C33) funded by MCIN/AEI /10.13039/501100011033 and by “ERDF A way of making Europe”. Also, to the Basque Government (projects TRUSTIND - KK-2020/00054, and REMEDY - KK-2021/00091). It is also funded by the QSALUD project (EXP 00135977/MIG-20201059) in the lines of action of the Center for Technological Development and Innovation (CDTI); and by the Ministry of Economy and Digital Transformation of the Government of Spain through the call for the Quantum ENIA project - Quantum Spain Project, and by the European Union through the Recovery, Transformation and Resilience Plan - NextGenerationEU in the framework of the Agenda España Digital 2025.

References

1. Aaronson, S.: BQP and the polynomial hierarchy. In: Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 141–150 (2009)
2. MacQuarrie, E.R., Simon, C., Simmons, S., Maine, E.: The emerging commercial landscape of quantum computing. *Nat. Rev. Phys.* **2**(11), 596–598 (2020)
3. Rojo, J., Valencia, D., Berrocal, J., Moguel, E., García-Alonso, J.M., Murillo, J.M.: Trials and tribulations of developing hybrid quantum-classical microservices systems. *arXiv*, vol. abs/2105.04421 (2021)
4. Moguel, E., Rojo, J., Valencia, D., Berrocal, J., Garcia-Alonso, J., Murillo, J.M.: Quantum service-oriented computing: current landscape and challenges. *Softw. Qual. J.* **30**(4), 983–1002 (2022)

5. Akbar, M.A., Khan, A.A., Mahmood, S., Rafi, S.: Quantum software engineering: a new genre of computing (2022). <https://arxiv.org/abs/2211.13990v1>
6. Garcia-Alonso, J.M., Rojo, J., Valencia, D., Moguel, E., Berrocal, J., Murillo, J.M.: Quantum software as a service through a quantum API gateway. *IEEE Internet Comput.* **26**, 34–41 (2021)
7. Piattini, M., Serrano, M., Perez-Castillo, R., Petersen, G., Hevia, J.L.: Toward a quantum software engineering. *IT Prof.* **23**(1), 62–66 (2021)
8. Endo, S., Cai, Z., Benjamin, S.C., Yuan, X.: Hybrid quantum-classical algorithms and quantum error mitigation. *J. Phys. Soc. Jpn.* **90**(3), 032001 (2021)
9. Schlosshauer, M.: Decoherence, the measurement problem, and interpretations of quantum mechanics. *Rev. Mod. Phys.* **76**(4), 1267–1305 (2005). <http://arxiv.org/abs/quant-ph/0312059>
10. Nachman, B., Urbanek, M., de Jong, W.A., Bauer, C.W.: Unfolding quantum computer readout noise. *npj Quantum Inf.* **6**(1), 1–7 (2020). <https://www.nature.com/articles/s41534-020-00309-7>
11. Georgopoulos, K., Emary, C., Zuliani, P.: Modelling and simulating the noisy behaviour of near-term quantum computers. *Phys. Rev. A* **104**(6), 062432 (2021). <http://arxiv.org/abs/2101.02109>. [quant-ph]
12. Arias, D., et al.: Let’s do it right the first time: survey on security concerns in the way to quantum software engineering. *Neurocomputing* **538**, 126199 (2023). <https://www.sciencedirect.com/science/article/pii/S0925231223003041>
13. Isakov, S.V., et al.: Simulations of quantum circuits with approximate noise using qsim and Cirq (2021). <http://arxiv.org/abs/2111.02396>. [quant-ph]
14. Gidney, C.: Stim: a fast stabilizer circuit simulator. *Quantum* **5**, 497 (2021). <http://arxiv.org/abs/2103.02202>. [quant-ph]
15. Aseguinolaza, U., Sobrino, N., Sobrino, G., Jornet-Somoza, J., Borge, J.: Error estimation in IBM quantum computers (2023). <http://arxiv.org/abs/2302.06870>. [physics, physics:quant-ph]
16. Romero-Álvarez, J., Alvarado-Valiente, J., Moguel, E., García-Alonso, J., Murillo, J.M.: Using open API for the development of hybrid classical-quantum services. In: Troya, J., et al. (eds.) *ICSOC 2022*. LNCS, vol. 13821, pp. 364–368. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-26507-5_34