Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

ARTICLE TYPE

Enabling Continuous Deployment Techniques for Quantum Services

Javier Romero-Álvarez[1] | Jaime Alvarado-Valiente[1] | Enrique Moguel[2] | Jose García-Alonso[1] | Juan M. Murillo[2]

[1]Quercus Software Engineering Group, Universidad de Extremadura, Cáceres, Spain

[2]Computing and Advanced Technologies Foundation of Extremadura, Cáceres, Spain

Correspondence
Javier Romero-Álvarez, Universidad de Extremadura, Avda. de la Universidad s.n., 10003 Caceres, Spain. Email: jromero@unex.es

Funding Information
This research was supported by the Grant PID2021-1240454OB-C31, TED2021-130913B-I00, PDC2022-133465-I00; QSALUD project (EXP 00135977 / MIG-20201059); the Ministry of Economic Affairs and Digital Transformation of the Spanish Government through the Quantum ENIA project call – Quantum Spain project, and by the European Union through the Recovery, Transformation and Resilience Plan – NextGenerationEU within the framework of the Digital Spain 2025 Agenda; and by the Regional Ministry of Economy, Science and Digital Agenda of the Regional Government of Extremadura (GR21133)

Abstract

Early advances in the field of quantum computing have provided new opportunities to tackle intricate problems in diverse areas as cryptography, optimization, and simulation. However, current methodologies employed in quantum computing often require, among other things, a broad understanding of quantum hardware and low-level programming languages, posing challenges to software developers in effectively creating and implementing quantum services. This paper advocates the adoption of Software Engineering principles in the field of quantum computing, thereby establishing a higher level of hardware abstraction that allows developers to focus on application development. With this proposal, developers will be able to design and deploy quantum services with less effort, similar to the facilitation provided by Service-Oriented Computing in the development of conventional software services. The present study introduces a Continuous Deployment strategy adapted to the development of quantum services, which covers the creation and deployment of such services. For this purpose, an extension of the OpenAPI Specification is proposed that allows the generation of services implementing quantum algorithms. The proposal was validated through the creation of an API with diverse quantum algorithm implementations, and evaluated through a survey of various developers and students who were introduced to the tool, with positive results.

KEYWORDS:
Quantum Computing, Quantum Services, Quantum Software Engineering, Continuous Deployment, OpenAPI

1 | INTRODUCTION

Quantum computing is a novel paradigm in computing. It replaces classical bits with quantum bits or qubits, enabling multiple simultaneous states for each bit, as well as entaglement between bits. These improvements offer an increased computational and data processing capacity, which has generated a growing interest in harnessing its potential for various areas of application[1].

Presently, quantum computers are used via cloud service providers; nevertheless, certain factors differentiate their usage in the cloud from the conventional utilization of classical computers[2]. On the one hand, quantum computers lack intricate operating systems capable of facilitating concurrent thread execution, as observed in classical systems. Consequently, when employing a quantum computer, task submissions are enqueued. And then, after fulfilling the request and executing the job,

specific procedures must be adhered to in order to obtain the outcome. These procedures are defined by each provider and may vary depending on the specific quantum service being used[3].

On the other hand, integrating quantum services and classical services is challenging due to various restrictions imposed by the nature of quantum systems. One such restriction is the superposition of states, where quantum particles can exist in multiple states simultaneously. This superposition is a fundamental aspect of quantum computing but poses challenges when integrating with classical services, which typically rely on deterministic states. In addition, the measurement of quantum states is probabilistic, which introduces uncertainty in the integration process[4].

To overcome these challenges when incorporating quantum services into existing systems, it is important to acknowledge the current existing limitations. Therefore, Quantum Software Engineering plays a critical role in the development and maintenance of quantum software systems by designing and implementing software solutions that effectively leverage the unique properties of quantum systems[5].

However, Quantum Software Engineering still has a long way to reach the level of classical software engineering. Challenges include hardware disparities and availability, as well as the need to use different programming languages for various quantum machine providers, specific libraries, and other details that depend on each provider[5]. Ongoing efforts aim to bring quantum computing in the cloud to the same standards as classical computing[6]. For instance, Amazon Braket offers a unified Software Development Kit (SDK) that enables developers to build quantum algorithms using a single programming language. It also allows algorithm testing on different simulators and execution on quantum computers from various providers, abstracting away any hardware differences. Other initiatives focus on mitigating hardware disparities and availability issues by providing methods to analyze and optimize quantum algorithms. These methods estimate in advance the most stable execution by considering the combination of hardware and quantum compilers[7].

Despite the progress made, there is still work to be done in the field of Service-Oriented Computing (SOC) within quantum computing, as it becomes more relevant especially because of its association with cloud computing[8]. However, there are still significant gaps between the understanding and application of SOC in classical computing and its application in the quantum domain. Presently, some endeavors are concentrated on bridging this gap by leveraging the advantages of SOC within the realm of quantum computing. Kumara et al.[9], for instance, has put forth a SOC-based methodology that offers a theoretical framework enabling both quantum and classical developers and programmers to create hybrid applications collaboratively. Nevertheless, the scarcity of available resources currently hampers the ability of developers to efficiently and extensively develop and deploy quantum services.

In the work presented in this paper, we focus on bringing the development of quantum services close to the development of classical cloud services by adapting existing tools and methodologies to support quantum services.

To this end, we propose a pipeline for the generation and deployment of quantum services by adapting techniques from the DevOps methodology for continuous software integration. In the field of quantum computing, the application of a DevOps process ensures the efficient development, testing and seamless deployment of quantum services, addressing the needs of both developers and operations teams. Specifically, we propose a modification of the OpenAPI Specification and its code generator to generate quantum services, as well as the automation of the Continuous Deployment (CD) process for their deployment in ready-to-consume containers. This is done through a DevOps-based workflow for continuous software integration and deployment, using the GitHub Actions[I] tool. To validate this workflow, a complete process has been developed through the following steps: an API, with diverse quantum algorithm implementations, has been specified for the services, the code for these services has been automatically generated with the OpenAPI extension, the services have been automatically deployed on an Amazon Web Services (AWS) server using the GitHub Actions tool and Docker[II], and their correct functioning has been manually verified by analyzing the generated code and making the necessary calls to the services. Also, in order to obtain a comprehensive evaluation of our approach, we conducted a survey involving developers and students who have utilized the pipeline and its associated tools.

The rest of the paper has the following structure. Section 2 analyzes the background of the present work and discusses the most relevant related works. Section 3 presents the proposed pipeline for quantum services generation and deployment, detailing the proposed process for the creation of quantum services. Section 4 demonstrates the feasibility of the generated services and deployment processes using several quantum algorithms and also evaluates the proposal through a survey of developers and students. Finally, Section 5 details the conclusions of this work.

[I] https://docs.github.com/en/actions
[II] https://www.docker.com/

2 | BACKGROUND AND RELATED WORK

In this section, we provide an overview of the key concepts and relevant background to our work, focusing on Service-Oriented Computing and DevOps methodology.

2.1 | Quantum Service-Oriented Computing

According to current trends, it is anticipated that complex quantum software systems will adopt a hybrid model, incorporating both quantum components and algorithms alongside classical software elements. A well-established approach for accommodating diverse components is through the utilization of SOC[10]. Extensive efforts are already underway in both the industry and research communities to employ SOC for the development of hybrid software systems[11]. The industry focuses on offering access to quantum computers through a Platform as a Service (PaaS) framework, while the research community concentrates on leveraging SOC for hybrid software system development[9].

However, creating and operating quantum service-oriented software remains a complex undertaking that significantly differs from the development of classical services, which professionals are accustomed to. The absence of advanced operating systems hinders the deployment of a quantum service in the same manner as a classical service. As an alternative, a classical service can be deployed to execute a quantum task upon invocation, introducing an additional layer of indirection and complexity to the system. Nonetheless, the limited abstraction level of most quantum programming languages and algorithms leads to heavy reliance on specific hardware for which the quantum source code is designed, thereby impeding developers from leveraging the availability of different quantum computers via cloud platforms[12].

The limited availability of advanced tools and methodologies for quantum service development leads to the utilization of low-abstraction and error-prone techniques by developers, which lack the benefits and advantages offered by modern software development tools. To tackle some of these challenges, in addition to the need to know each specific language and particularities of each quantum provider, various research initiatives are emerging.

Weder et al.[13] have performed a study focusing on issues associated with orchestrating hybrid systems that involve both quantum and classical services. To address these concerns, they propose the adoption of a TOSCA-based orchestration mechanism to coordinate the different services.

A similar study by Garcia-Alonso et al. in[14] introduces the concept of a Quantum API Gateway, which dynamically determines the most suitable quantum computer for executing a specific quantum service. These researchers have also explored the key problems of quantum services[15] and presented a deployment guide for such services[16].

Furthermore, there are emerging endeavors in the field of hybrid quantum-classical computing that aim to provide Application Programming Interfaces (APIs). Examples of such efforts include Qiskit Runtime[17] and Quantum Intermediate Representation (QIR)[18]. These initiatives are designed to enable users to efficiently execute workloads while serving as a standardized interface between different programming languages and quantum computing platforms.

Although these proposals elevate the level of abstraction in quantum service development and simplify the creation of intricate hybrid solutions, where classical and quantum services coexist, they still fall short compared to the classical development of service-oriented software. To address this limitation, additional support is required to facilitate the creation of quantum services. This support would enable current service developers to transition more easily into the quantum domain, thereby mitigating the shortage of skilled quantum workforce and fostering the development of a new generation of hybrid systems[19].

2.2 | Leveraging Continuous Deployment for Quantum Software

DevOps is a software development methodology that seeks to integrate and synchronize the efforts of development and operations teams, aiming to optimize the software delivery process[20]. It places particular emphasis on fostering collaboration, implementing automation, and enhancing communication channels in order to enhance the velocity, quality, and dependability of software deployment.

In the field of quantum software development, DevOps assumes significant relevance by addressing the distinctive challenges inherent to quantum computing. For instance, due to the presence of quantum noise-induced errors, quantum software needs stricter testing and validation protocols compared to classical software. DevOps can facilitate the automation of these testing and validation procedures, thereby enhancing the dependability of quantum software[21]. Furthermore, quantum software tends

to exhibit heightened complexity and computational demands compared to classical software. DevOps can alleviate this complexity by automating the deployment, scaling, and maintenance aspects of quantum software, thus augmenting its efficiency and scalability throughout the development lifecycle[22].

Continuous Integration (CI) and Continuous Deployment (CD) represent fundamental practices within the DevOps methodology that seek to automate the software delivery process. CI entails the integration of code modifications into a shared repository, while CD involves the automated deployment of software updates to the production environment promptly upon their completion[23].

Presently, there are a few approaches for integrating quantum algorithms into the various development stages of a DevOps cycle. An example of applying continuous deployment principles can be seen in the work by I.D. Gheorghe-Pop et al.[21], where they propose an adaptation of the traditional DevOps process to the quantum realm, termed Quantum DevOps. This research presents a significant contribution by introducing and advocating for the concept of Quantum DevOps, emphasizing its potential benefits and importance.

These proposals account for quantum computing characteristics such as qubit quantity, cloud accessibility, and the error rates of individual quantum machines. However, no tools or implementations have been developed to aid developers in this process. Consequently, despite the existence of proposals to incorporate quantum algorithms into CD stages, certain aspects of quantum CD remain unaddressed by these aforementioned proposals. In particular, critical phases in these cycles, such as build and deployment, require consideration. Build and deployment represent crucial stages involving the creation of the software package and its subsequent deployment into the production environment. This paper aims to encompass these aspects and optimize the generation and deployment processes, thus expediting the delivery of quantum computing applications.

3 | PIPELINE FOR QUANTUM SERVICES GENERATION AND DEPLOYMENT

This section provides a comprehensive explanation of the pipeline proposal for the generation and deployment of quantum services, making use of OpenAPI, Docker, and GitHub Actions. OpenAPI—which has been modified to support quantum services—offers a standardized specification for the development and generation of APIs, while Docker facilitates the creation and deployment of containers capable of deploy these APIs. By combining these technologies, we propose a flexible and robust approach to the creation, management, and deployment of quantum services.

The complete pipeline is performed using GitHub Actions to automate the process, enabling CD of quantum services. GitHub Actions is a feature provided by GitHub that allows developers to automate various tasks and workflows within their software development process. It acts as a continuous integration and deployment (CI/CD) tool, which means it helps in automating the building, testing, and deploying of code changes. In the context of this work, GitHub Actions is used to automate code generation and deployment for quantum services. It integrates with the modified OpenAPI Specification to facilitate a seamless transition from manual processes to automated workflows.

The proposed CD pipeline is represented in Figure 1 as a Business Process Model and Notation (BPMN) diagram.

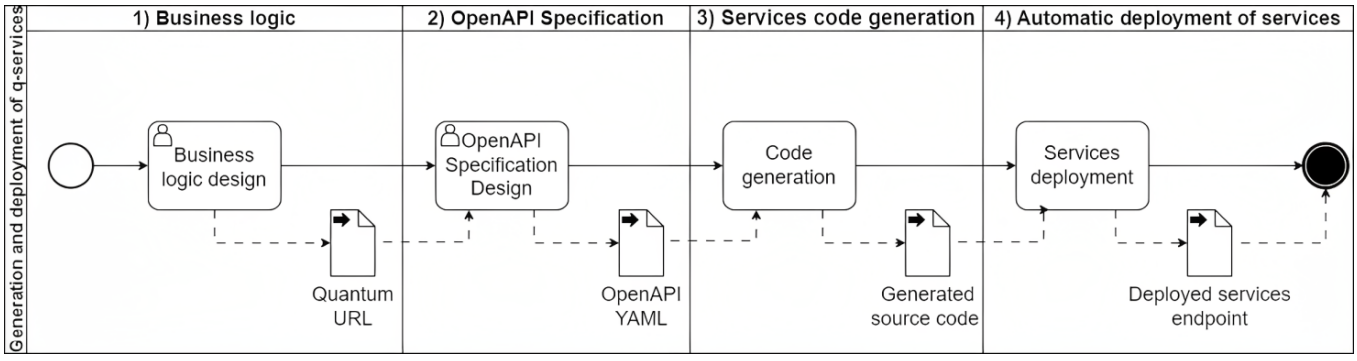


FIGURE 1 CD pipeline for quantum services generation and deployment

The implementation of this CD pipeline can be accessed in the following repository[III]. It includes all the essential components and configurations necessary for enabling the seamless deployment of quantum services.

In the following sections, we will discuss the four main phases of the proposed pipeline (Figure 1) and its benefits over traditional methods. Section 3.1 elaborates on the process of defining the business logic of quantum services in the form of circuits. Subsequently, Section 3.2 expounds on how these circuits, in conjunction with the OpenAPI Specification, enable the definition of the services. Next, Section 3.3 explains the subsequent generation of the services code based on the aforementioned specification. And finally, Section 3.4 outlines the automatic deployment of these quantum services.

All the steps of this process will be explained taking as an example a service that implements Shor's algorithm, one of the best known algorithms in quantum computing.

3.1 | Business logic

The initial phase of the pipeline implies the definition of the business logic for the quantum service in the form of a quantum circuit—as shown in Figure 2.

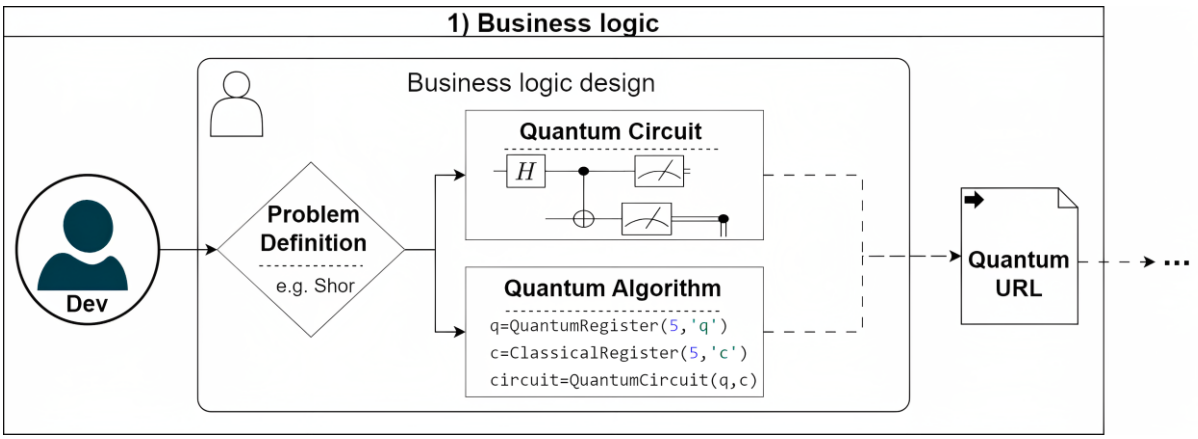


FIGURE 2 Definition of the business logic

To integrate the business logic of quantum services seamlessly, we leverage a graphical quantum programming tool that facilitates the construction of quantum circuits through intuitive drag-and-drop operations. Specifically, the web-based quantum programming tool called Open Quirk[IV].

Developed as an open-source software solution utilizing the JavaScript programming language, Open Quirk streamlines the rapid designing of quantum circuits. Moreover, it provides programmatic access to the composed quantum circuits via its graphical editor.

It is worth noting that the utilization of Open Quirk can be substituted with any other quantum circuit creation tool that offers programmatic access to the underlying code, thereby ensuring compatibility with the OpenAPI Specification.

Furthermore, as an alternative to employing a graphical circuit editor, we offer the option of directly incorporating quantum circuits into the service endpoint through a URL of the repository hosting the Python file with the source code of the circuit, such as Qiskit code.

The URL containing the quantum circuit, whether it be in the form of a circuit constructed in Quirk or an already implemented circuit, serves as the input for the subsequent phase of the pipeline. Using the example of Shor's algorithm, this URL[V] provides the implementation of the circuit, and here is the link[VI] to view its design in Open Quirk.

[III] https://github.com/javierrome236/quantumDeployment
[IV] https://algassert.com/quirk
[V] https://bitbucket.org/spilab/quantum-circuits-code/raw/7f65dcd5f59bf75108234bfc5234e54247b3916f/Shor_algorithm.py
[VI] https://shorturl.at/anuQ1

3.2 | OpenAPI Specification

To successfully implement a classical service within the OpenAPI Specification, developers must consider two crucial aspects: the business logic, which encompasses the service's functionality, and the service's endpoint, which governs how external clients can interact with the service.

The OpenAPI Specification plays a pivotal role in defining the service's endpoint, employing a language-independent, standardized interface tailored for RESTful APIs. This standardized approach allows users to explore and understand the service's capabilities without requiring direct access to the source code or extensive documentation. It enables the definition of the service itself, including the specification of input and output parameters. Leveraging a source code generator, the code structure is then automatically generated based on this specification—which is explained in the next phase of the pipeline. Developers can select their preferred programming language, and subsequently integrate the business logic into the generated code, resulting in a fully functional service[24,25]. Furthermore, from a scientific standpoint, OpenAPI enables the smooth exchange of information between diverse software systems by establishing a common vocabulary to describe API functionality.

In this attempt to address the current state of the art in quantum services, we adopt a similar approach to that employed for classical service implementation, utilizing OpenAPI. To achieve this, an extension of the OpenAPI Specification has been devised, encompassing custom properties tailored specifically for defining quantum applications.

Specifically, the generated OpenAPI Specification is extended with custom properties to incorporate the quantum aspects of the service, such as the URL obtained in the previous step—Figure 3. This extension allows the definition of quantum applications with OpenAPI. By integrating the specific properties of quantum computing in the OpenAPI specification, we pave the way for the seamless integration of quantum services into the broader service ecosystem.

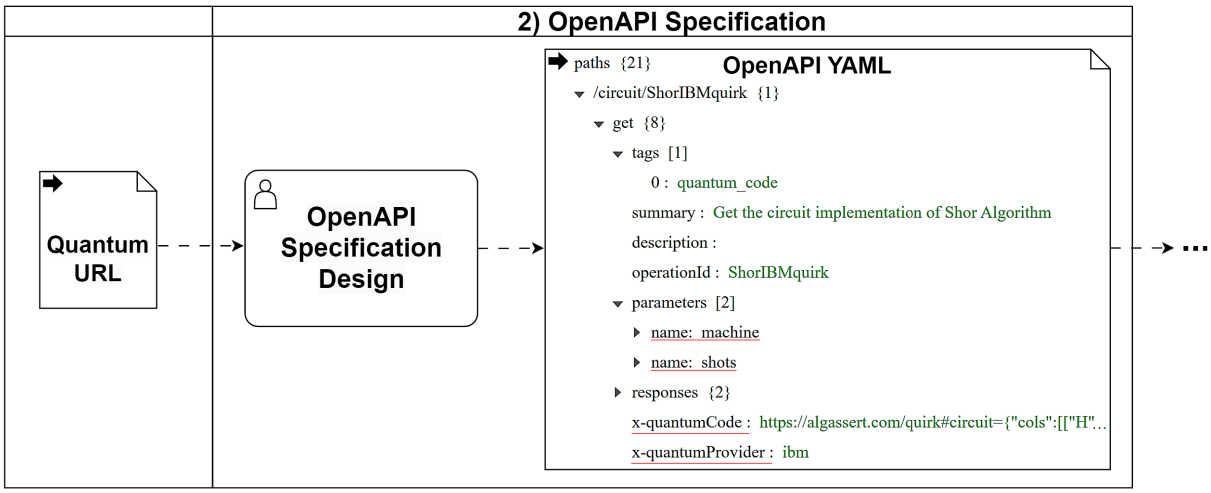


FIGURE 3 OpenAPI Specification for quantum services

In this study, we propose the utilization of custom properties, also known as specification extensions or provider extensions, to extend the OpenAPI Specification and enable the inclusion of quantum circuits. These custom properties serve as a means to incorporate supplementary information into the API contract definition, expanding the scope of the specification.

In Figure 3 we also can see a fragment of the YAML API contract, with the service endpoint of the Shor's algorithm, showcasing some of the custom properties and parameters added to the specification. This YAML contract containing custom quantum properties, quantum provider-specific details, and the URL to the circuit, is the input to the next phase of the pipeline.

Next, we describe briefly the most important properties and parameters. The first custom property, named *x-quantumCode*, contains the URL that hosts the implemented circuit code or Open Quirk URL. The second custom property, named *x-quantumProvider*, allows developers to designate the service provider on which the quantum service will be executed. Presently, it allows the selection between two prominent service providers: IBM Quantum, which provides the Qiskit Development Kit, and Amazon Braket, which offers a suite of quantum computing services. We use these two providers because they are two of the most widely used today and cover a wide range of languages and hardware vendors.

Additionally, two parameters, namely *machine* and *shots*, have been incorporated into the specification to ensure the proper execution of the quantum services. By focusing on these key parameters, the OpenAPI Specification remains concise and targeted, providing the necessary flexibility and control for users to interact with quantum services effectively. The decision not to add additional parameters, beyond *machine* and *shots*, in the OpenAPI Specification is based on prioritizing the accurate representation of quantum functionality and maintaining compatibility with the standardized approach provided by OpenAPI

The *machine* parameter allows the service client to dynamically select the specific machine on which the quantum algorithm will be executed, choosing from the available options offered by the service provider specified in the custom *x-quantumProvider* property. This flexibility is necessary as Amazon and IBM provide different quantum computers and simulators to developers.

Moreover, the *shots* parameter allows specifying the desired number of times the circuit should be executed on the chosen quantum machine for each service request. This parameter is essential for controlling the statistical sampling and obtaining reliable results from quantum algorithms.

3.3 | Services code generation

In the third phase of the pipeline, the extension proposed for the OpenAPI Code Generator[VII] is employed to automatically generate the source code for the quantum services, from the YAML specification of the previous step —as shown in Figure 4.

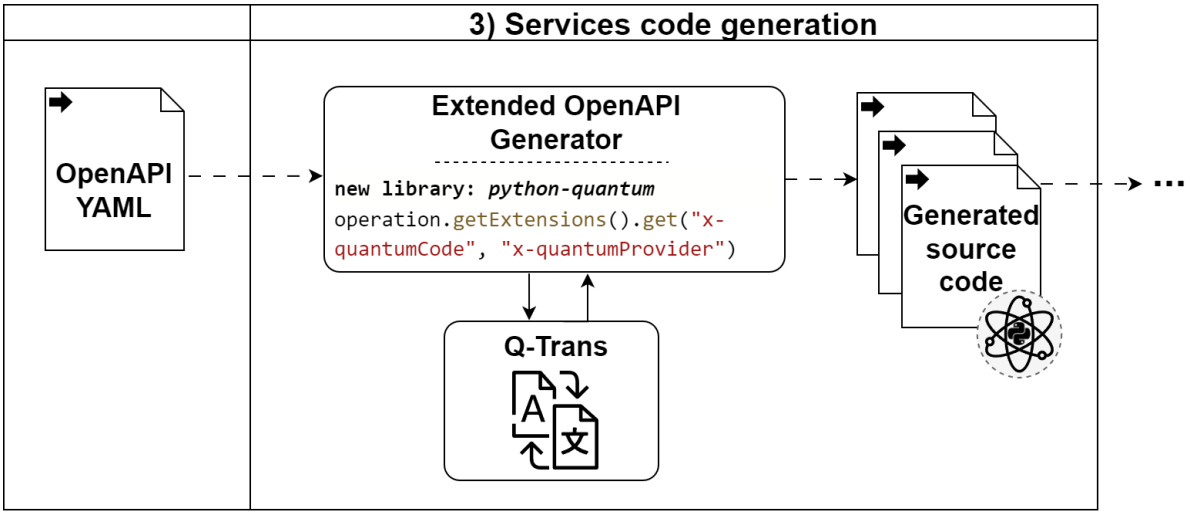


FIGURE 4 Quantum services code generation

The OpenAPI Code Generator, developed by the OpenAPI initiative, is a web tool that enables the creation of server applications and client APIs based on an OpenAPI Specification. Comprising various modules and libraries, this code generator facilitates code generation for multiple programming languages. In our work, we have extended this tool by introducing a new library called *python-quantum*, which leverages the existing Python library for API generation, namely Flask[VIII]—a lightweight and widely adopted Python web application framework. Consequently, the generated source code implements quantum API endpoints in Python using Flask. These endpoints are responsible for deploying and executing the designated quantum algorithm on the provider and machines specified by the developer, utilizing the custom properties incorporated into the OpenAPI Specification, explained above.

In this way, these endpoints contain the necessary source code for executing a quantum task on the designated service provider, as indicated by the custom property *x-quantumProvider*. The quantum task's source code is also generated for the specified provider, employing the quantum circuit specified in the *x-quantumCode* property.

[VII] https://openapi-generator.tech/
[VIII] https://flask.palletsprojects.com

Depending on which provider the service is deployed with, it is necessary to generate the code in the specific language of that provider. Therefore, a previous translation process is necessary for those services where the logic has been defined through Quirk, and not with Python code.

For this process, we have developed a translator API—named *Q-Trans*—, to perform the translation accurately and efficiently, and to handle the complexities of quantum programming languages more proficiently[IX]. To this end, the *Q-Trans* tool features an endpoint for each supported provider. These endpoints receive the Open Quirk URL, format it accordingly, and generate the code utilizing the specified gates in the respective programming language. The quantum machine in which the algorithm will be executed is provided as a parameter by the service client.

The generated source code encompasses the essential libraries of the quantum provider, a classical wrapping service, and the supported machines available at the quantum provider. Additionally, it incorporates the business logic derived from the quantum circuit, translated into the language supported by the quantum provider. In the following repository[X] is available the code of the generated service that encapsulates Shor's algorithm circuit.

The resulting quantum services can be deployed and invoked in the subsequent phase of the pipeline by making RESTful calls to the endpoints that encapsulate the quantum algorithm code.

3.4 | Automatic deployment of services

Once the source code of the quantum services has been generated, in this final phase we focus on automating its deployment. This automation involves ensuring that the services, which were automatically generated in the previous phase, are subjected to a systematic deployment process facilitated by CD, using software containers—as shown in Figure 5.

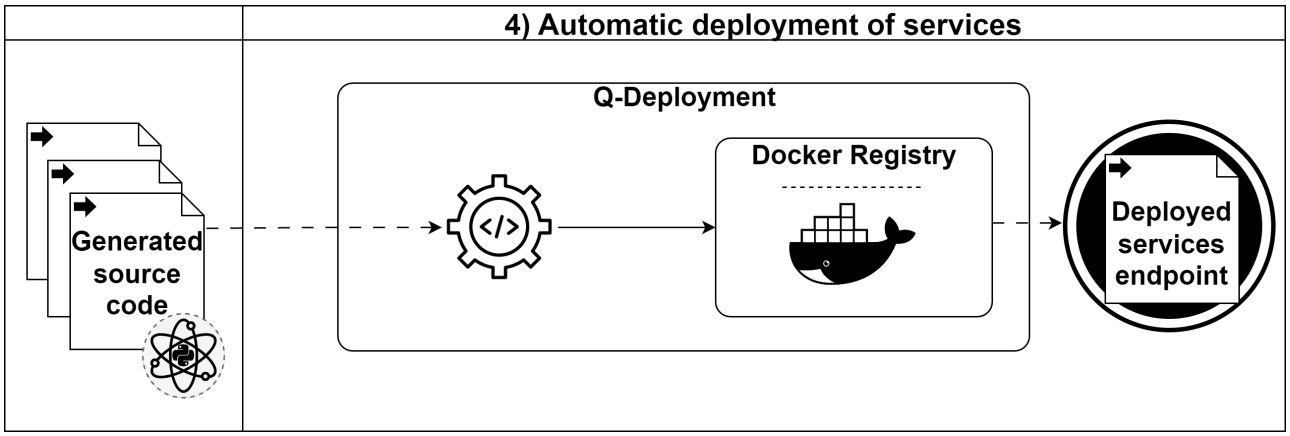


FIGURE 5 Automatic deployment of services

Containers are a useful solution for isolating software components individually and configuring multiple deployments. For this, the Docker development tool has been used, which is an open-source platform that allows the deployment of applications within software containers, and which also allows the process to be automated.

The tool developed for automating the creation and deployment of containers in the context of this project is referred to as *Q-Deployment*. This tool uses the generated source code to encapsulate it in the Docker container, and then deploy it.

Also, this tool and GitHub Actions are related in the sense that they both play a role in automating the creation and deployment of containers for quantum services.

Thus, the source code generated for the quantum services is encapsulated within a Docker Registry container. This encapsulation process utilizes a pre-defined *Dockerfile* that incorporates the necessary instructions for appropriately preparing the container for deployment.

[IX] https://github.com/JaimeAlvaradoValiente/openapi-generator-quantum
[X] https://github.com/javierrome236/quantumDeployment/suites/13268674896/artifacts/736152926

To enhance the replicability of the system, a foundational registry container has been proposed and made available within the repository[XI]. This container is built upon a Python image and is equipped with all the requisite dependencies for seamless execution. By leveraging this pre-configured container, the time-consuming process of container creation is significantly expedited.

Finally, the deployment of the container is exposed through one of the server's ports. To facilitate this, the system checks for available ports from a pre-defined list and assigns the first available port. It is important to note that, in the present context, the services can only be deployed on the same server where the entire system is deployed. Subsequently, the user is furnished with a link to the server, appended with the designated port number, providing access to the generated services. In the following link[XII] is deployed the Shor service, from the example used throughout the previous steps.

Overall, the presented pipeline demonstrates a systematic and automated approach for the generation and deployment of quantum services within a containerized environment. By leveraging GitHub Actions and the modified OpenAPI Specification, developers are able to seamlessly transition from manual processes to automated code generation and deployment.

Further details on the visualization and execution of the services can be found in Appendix A, for a comprehensive description of the service display and deployment procedures.

4 | VALIDATION

In this section, we validate the complete pipeline for the CD of quantum services. Section 4.1 presents the validation of the tool by generating and deploying quantum services containing 40 real quantum algorithms. Section 4.2 below evaluates the proposed pipeline through a survey of different developers and students who were introduced to the tool.

4.1 | Generating quantum services with quantum real algorithms

In this paper, we present a procedural process for the automated generation and deployment of quantum services, with a specific focus on the CD process. The efficacy of this pipeline has been evaluated through the creation of an API encompassing multiple services, each encapsulating diverse implementations of quantum algorithms.

4.1.1 | Implementations of quantum algorithms

To verify the functionality and deployment of the proposed pipeline, we have employed a range of gate-based quantum circuit algorithms representing various quantum algorithms. These algorithmic implementations cover a wide range of applications, including both quantum-classical hybrids and purely quantum algorithms.

The quantum algorithms utilized for this validation process can be accessed from a publicly available repository[XIII], wherein 20 of them have been implemented in Python code, while the remaining 20 have been implemented using the quantum circuit composer Open Quirk. Consequently, the API consists of a total of 40 distinct services, as each implementation has been generated for both AWS and IBM platforms, denoting the corresponding providers within the OpenAPI Specification.

Among the algorithms employed, we highlight the utilization of Grover's algorithm[26], a quantum search algorithm renowned for its ability to search an unsorted database of N items in $O(\sqrt{N})$ time, as opposed to the $O(N)$ time complexity of classical algorithms. This algorithm finds applications in tasks such as database search and pattern matching.

Another algorithm of significance is Shor's algorithm[27], which surpasses the efficiency of classical algorithms in integer factorization. Shor's algorithm demonstrates exponential speedup and carries paramount importance in the field of cryptography, as the security of widely used public-key cryptosystems, like RSA, relies on the computational infeasibility of factoring large integers.

Additionally, Simon's algorithm[28], an algorithm that exploits quantum superposition to detect hidden patterns in oracle outputs, has been integrated into our framework, further underscoring its versatility in quantum computing.

To assess the generation and deployment process, we also incorporate the implementation of the Quantum Approximate Optimization Algorithm (QAOA)[29]. This class of algorithms finds applications in solving optimization problems within domains such as finance and engineering.

[XI] https://hub.docker.com/r/jromero236/quantumservices
[XII] http://quantumservicesdeployment.spilab.es:8082/ui/
[XIII] https://bitbucket.org/spilab/quantum-circuits-code

Lastly, our collection of algorithmic implementations, which is summarised in Table 1, includes other circuits employed as subroutines in quantum computing.

Number	Circuit
1	Shor's algorithm
2	Bernstein-Vazirani
3	Grover's algorithm
4	Deutsch-Jozsa Algorithm
5	Simon's algorithm
6	Travelling Salesman Problem (TSP)
7	Quantum Teleportation
8	Quantum Phase Estimation
9	Quantum Fourier Transform (QFT)
10	Quantum Approximate Optimization Algorithm (QAOA)

TABLE 1 Summary of the used quantum circuits

Through comprehensive evaluation of these algorithmic implementations, we substantiate the efficacy and viability of our automated approach for the generation and deployment of quantum services. These findings lay a strong foundation for further research and development endeavors in this domain, propelling the practical applications of quantum computing forward.

4.1.2 | Results of the deployments

The proposed approach for the automatic generation and deployment of quantum services has been evaluated to assess its effectiveness. In order to determine the correctness of the deployment, several aspects were considered. Firstly, it was essential to verify that the circuit code was accurately encapsulated within the service, including all the necessary libraries for its execution. Secondly, the functionality of the API needed to be validated to ensure that it was functioning correctly. Finally, the encapsulated quantum circuits were executed and evaluated for any errors, with a focus on whether they produced the expected results.

To test the last aspect, the different services were subjected to experiments by running the circuits on multiple quantum machines provided by AWS and IBM. The aim was to assess whether the deployed services operated as intended and were capable of delivering the desired outcomes. Access to the correctly generated and deployed services was made available through the following direction[XIV].

The experiments involved three different methods of generating the circuits: two using Qiskit and Amazon Braket and the other involving circuits generated by Quirk Composer—a total of 40 services. The results revealed that all 20 quantum services utilizing Qiskit and Amazon Braket generated circuits respectively were successfully deployed. However, when it came to services containing circuits generated by Quirk Composer, it was observed that 17 out of 20 were deployed correctly, while 3 experienced deployment failures—Figure 6.

Upon conducting a more thorough analysis, it was discovered that one of the potential causes for the failed deployments was the limited support for specific circuit gates, particularly phase gates, within the code generator employed by the Q-Trans. This limitation prevented the code generator from effectively handling and incorporating all types of circuit gates present in the quantum circuits.

Quirk-specific phase gates, in which an angle can be specified, were not properly translated or included in the generated code, resulting in errors during the deployment process. For example, the phase gates used in the QAOA circuit [XV]. As a result, the affected quantum services utilizing circuits with phase gates encountered deployment failures. To overcome this issue, a new version of the code generator is currently being developed. The aim is to enhance its capabilities and ensure seamless integration and translation of all circuit gates, including phase gates and other gates that have not been contemplated. By addressing this limitation, the new version of the generator is expected to facilitate successful deployments of quantum services containing circuits with a broader range of gate types.

[XIV] http://quantumservicesdeployment.spilab.es:8082/ui/
[XV] https://shorturl.at/myPX8

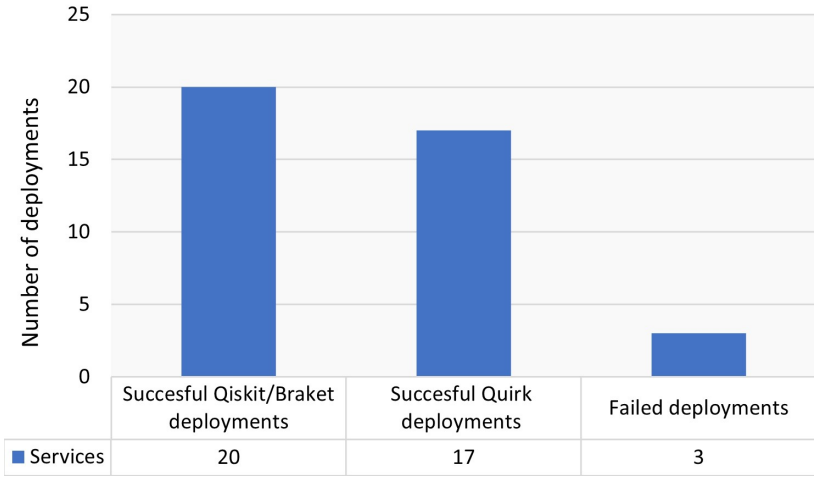


FIGURE 6 Number of failed and successful deployments

Overall, the proposed approach demonstrated a promising success rate for automatic quantum service generation and deployment. These results highlight the potential of the proposed approach to significantly reduce the time and effort required for circuit generation and deployment. Further research can be conducted to identify the factors contributing to the deployment failures and to explore possible solutions to mitigate them.

4.2 | Evaluation of the automatic quantum deployment pipeline

The proposed pipeline for the Continuous Deployment of quantum services has been also evaluated by developers and students. This validation consisted of a training course that aims to help software engineers to address the difficulties encountered when developing quantum services.

A total of 62 people participated in the training course, which was held at two universities—the University of Extremadura[XVI] and the University of Málaga[XVII]—as well as at the Bertinoro International Spring School[XVIII] and as a tutorial at the International Conference on Web Engineering[XIX].

The course introduces basic concepts of quantum computing and shows the structure of a quantum program and the main techniques and patterns currently used to develop them. Subsequently, the use of the proposed pipeline to develop the same programs is shown. At the end of the course, we conducted a survey to find out about the usefulness of the tool presented.

4.2.1 | Problems and questions

As mentioned in previous sections, several problems related to the development and deployment of quantum services have been detected, such as hardware differences and availability, and lack of tools and methodologies to support the development of quantum services on par with classical cloud services. To this end, we propose a pipeline for generating and deploying quantum services based on CD.

These issues led us to conduct a survey to collect feedback from developers/students on their experience with existing methods and with this proposed approach. The questions on which we based the survey are as follows:

- **General questions** have been posed to obtain a profile of the survey participants. Such as their employment status, academic background, or their experience in the field of quantum computing.
- **Technical questions** have been posed to analyze the current perception of quantum software development in terms of the aforementioned problems and mainly to corroborate the easiness and usability of the process presented:

[XVI] http://qserv.spilab.es/curso-uex-2023-introduccion-al-desarrollo-de-software-cuantico/
[XVII] https://www.ensenanzaspropias.uma.es/informacion_curso.php?id_curso=6903577
[XVIII] https://tempesta.cs.unibo.it/projects/BISS/2023/courses/#hybrid-quantum-computing
[XIX] https://icwe2023.webengineering.org/quantum-web-services-development-and-deployment/

- **RQ1. In your opinion, what is the main aspect of Quantum Computing that should be improved from a software engineering point of view?** This question was raised to obtain the perspective and opinions of respondents on specific areas for improvement at the intersection of quantum computing and software engineering.
- **RQ2. Have you designed any quantum circuits and executed them in a quantum machine or simulator before?** This question aims to evaluate the level of knowledge and practical experience of the respondents in the field of quantum computing. Specifically in the design and implementation of quantum circuits.
- **RQ3. Have you developed or deployed any quantum services before?** This question is related to the previous one, but in this case, the answers to this question will provide information on the familiarity of the respondents with the development of applications and services in the field of quantum computing.
- **RQ4. How challenging do you find the process of generating and deploying quantum services using current methods?** This question seeks to assess the degree of complexity and difficulties perceived by respondents in the development and implementation of quantum services using existing methods seen in the course.
- **RQ5. Is it essential for you to have tools that will allow the development and deployment of quantum services?** This question is useful to know if the respondents see the need for tools that allow in terms of facilitating and facilitating the development process of quantum services.
- **RQ6. Have you understood the concepts explained in the course?** This question allows us to know if the contents explained during the course have been understood or not, with a view to future improvements both in the proposed process and in the course.
- **RQ7. How much easier do you think it would be to develop and deploy quantum services using the semi-automatic method presented compared to the current methods?** This question allows us to evaluate the opinion of the respondents on the improvement degree that the presented semi-automatic approach can offer in terms of simplicity and efficiency in the development and deployment of quantum services.
- **RQ8. How effective do you think the Continuous Deployment (CD) approach would be in simplifying and facilitating the development process of quantum services?** Responses to this question will provide information on the respondents' expectations and perceptions of the potential benefits and advantages of the Continuous Deployment approach compared to existing methods.
- **RQ9. How important is it for you to have a simplified workflow for generating and deploying quantum services?** This question allows us to determine the importance that respondents assign to having a simplified workflow for generating and deploying quantum services.

Appendix B contains all questions, possible answers, and obtained results. The analysis of the results in the following subsection highlights the most crucial findings.

4.2.2 | Main results of the survey

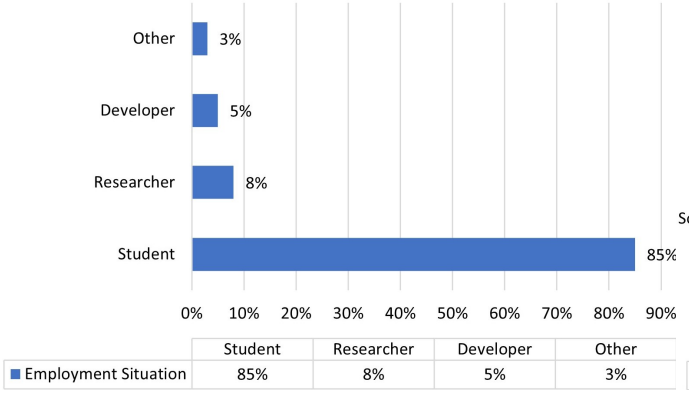
With respect to the main results of the general questions, Figures 7a and 7b show the percentages of the 62 participants' profiles in the survey. Being 51% computer scientists, 85% students, and practically all with few years of experience in quantum computing.

Looking at the results with respect to the research questions related to the current perception of quantum software development, with the responses of RQ1 we can conclude that the main aspect of quantum computing that should be improved from a software engineering perspective is the developer experience. Participants highlighted the need for more accessible programming languages and tools, improved availability and accessibility of quantum machines, standardization of access across vendors, and increased levels of abstraction in quantum computing.

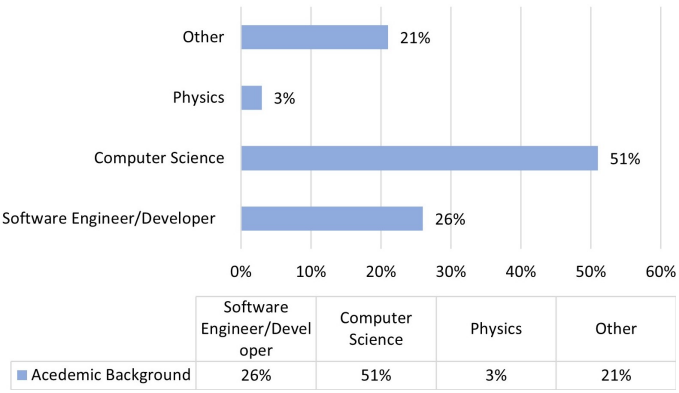
For RQ2—Figure 8—, the majority of participants (87%) have designed quantum circuits but have not executed them on real machines. This is related to RQ3—Figure 9—, where 97% of the participants have not developed or deployed any quantum services before, but they have worked in quantum computing.

Related to the lack of tools and methods for the development of quantum services, the responses from RQ4—Figure 10—reflect that the process of generating and deploying quantum services using current methods is considered challenging by respondents. 38% find it somewhat challenging, while 44% find it very challenging. This indicates significant difficulties associated with the current methods.

Finally, in RQ5 —Figure 11—, a majority of respondents (almost 80%) consider it essential to have tools that enable the development and deployment of quantum services. This aligns with the high percentage of respondents finding the current process challenging, indicating a strong demand for improved tools and methods.



(a) Current employment situation



(b) Academic background

FIGURE 7 Results obtained from participants' profiles

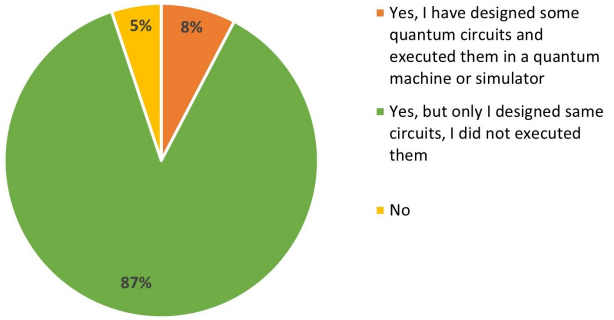


FIGURE 8 Results of RQ2

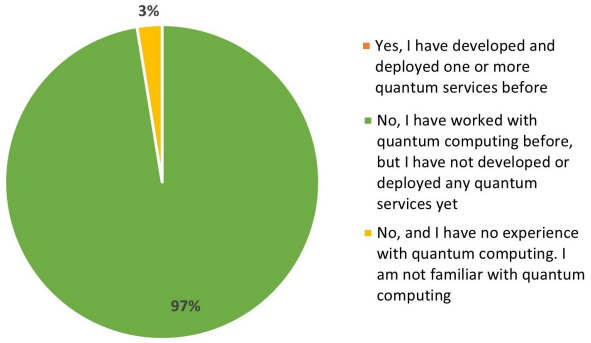


FIGURE 9 Results of RQ3

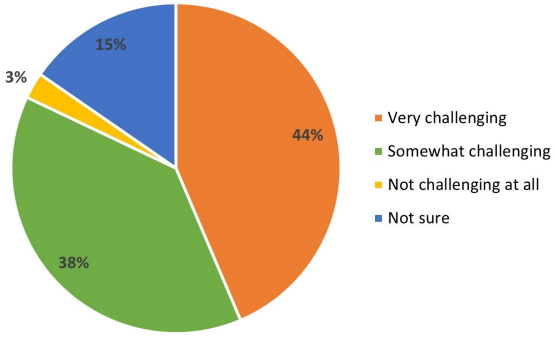


FIGURE 10 Results of RQ4

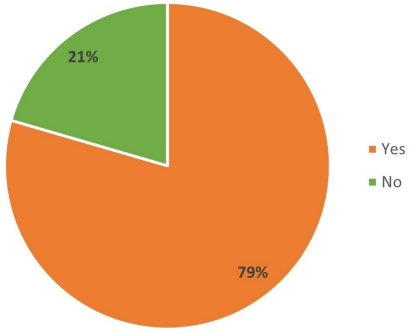


FIGURE 11 Results of RQ5

In summary, the results of this part of the survey highlight the importance of improving the developer experience in quantum computing. Participants expressed the need for accessible programming languages, better access to quantum machines, standardization, and increased abstraction. While most participants have designed quantum circuits, they have not yet deployed quantum services. The current process of generating and deploying quantum services is considered challenging. These findings emphasize the significance of developing tools and methods to facilitate the development and deployment of quantum services, as expressed by the majority of respondents.

Regarding the results related to the questions that specifically evaluate the usefulness of the proposed process, we have obtained quite positive results that validate our proposal.

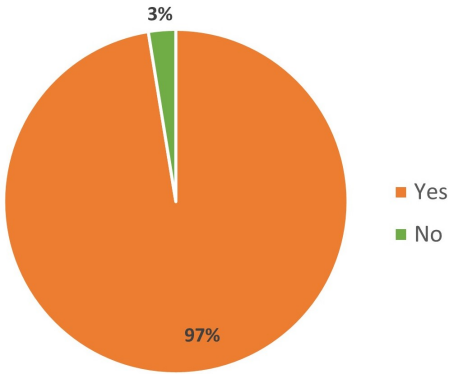


FIGURE 12 Results of RQ6

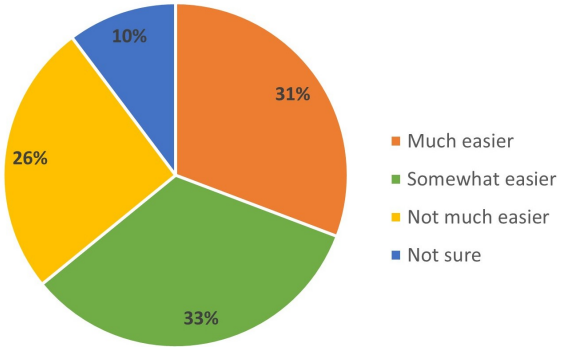


FIGURE 13 Results of RQ7

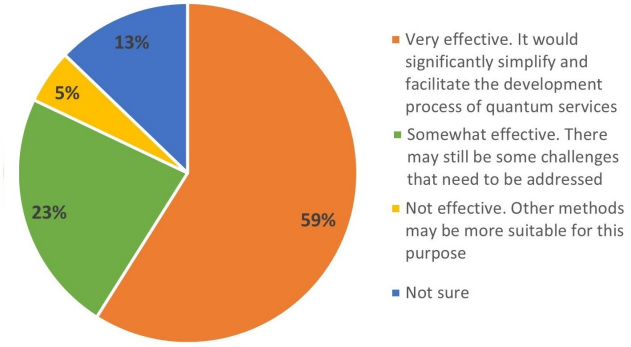


FIGURE 14 Results of RQ8

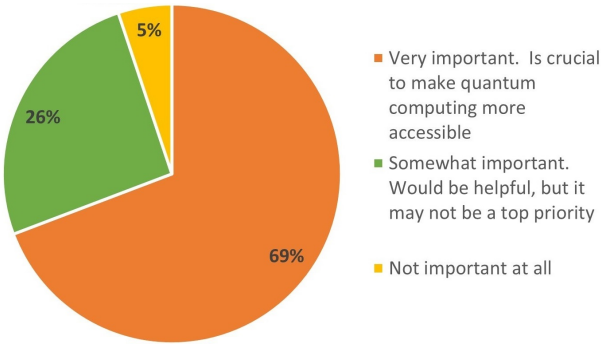


FIGURE 15 Results of RQ9

First, Figure 12 shows the results of RQ6, detailing that the majority of participants (more than 97%) clearly understood the concepts explained during the course and learned to use the proposed tool. This validates that our process is easy to use even for developers with low experience in quantum computing.

Furthermore, regarding the results of question RQ7—as shown in Figure 13—, a total combined percentage of 64% (33% + 31%) of the participants believe that using semi-automatic methods would make the process either "*much easier*" or "*somewhat easier*". This suggests that a significant portion of the participants perceive semi-automatic methods as beneficial for simplifying the development and deployment of quantum services. Additionally, 10% of the respondents are unsure about the impact of semi-automatic methods. This suggests that further research, development, and user feedback are needed to assess the true efficacy and feasibility of the proposed semi-automated methods in the context of quantum services development and deployment.

On the other hand, RQ8 also validates our proposal for the CD of quantum services—Figure 14—, where the majority of the respondents (59%) perceive a CD approach as "*very effective*" in significantly streamlining and facilitating the development process of quantum services. This suggests that participants recognize the potential benefits of the CD process proposed, in terms of improving efficiency, reducing manual effort, and ensuring rapid iterations and updates in the development of quantum services. Although, there may still be challenges or limitations that need to be addressed, as 23% of the participants believe.

Finally, as the results of question RQ9 reflect in Figure 15, the majority of participants (69%) consider it very important to have a simplified workflow for generating and deploying quantum services. This indicates a strong belief that simplifying the process is crucial to making quantum computing more accessible and practical for developers. Participants recognize the challenges and complexity associated with quantum computing and believe that the proposed workflow simplification can reduce barriers and enable more efficient development and deployment of quantum services.

5 | CONCLUSIONS

This research has identified several existing constraints in the creation and utilization of quantum services, highlighting the potential for adapting classical service engineering techniques and methods to the quantum realm. Consequently, developers

experienced in high-level abstraction tools that streamline the development and deployment process of classical services can expect a smoother transition to quantum service development.

To this end, we have proposed a method to standardize the process of defining quantum services using the OpenAPI Specification and we have provided an extension of the OpenAPI Code Generator capable of generating the source code of quantum services from an API specification and a quantum circuit. To facilitate the automatic deployment of these quantum services, we have developed a workflow for the Continuous Deployment of the code generated in Docker containers using the GitHub Actions tool. During this process, it checks if the YAML format of the specification is correct, calls the code generator, encapsulates the code in a container, and finally returns to the user the URL where the services are deployed.

By leveraging GitHub Actions along with other tools like *Q-Trans* and *Q-Deployment*, which are contributions of this work, developers can streamline their development process by automatically generating source code for quantum services and deploying them within containerized environments.

With our proposal, we aim to make quantum computing more accessible by offering a tool for the automatic generation and deployment of quantum services. By automating the deployment process, our tool brings quantum computing closer to what we have of classical computing.

To validate the proposal, we have generated 40 services that encapsulate 20 quantum circuits. There are 20 of them implemented in Qiskit and in Amazon Braket and another 20 using the quantum circuit composer Open Quirk, which are available in a public repository. We have executed the pipeline with all the services and have successfully deployed 37 of them (92.5% success), with failed deployments being the cause of limited support for specific circuit gates.

As shown by surveys carried out and explained in this paper, there are few tools that allow a higher level of abstraction when developing quantum software and more than 70% of respondents consider it essential to have tools that facilitate development using the architectural style proposed in this paper. In conclusion, we consider it essential to have tools that facilitate development using the proposed architectural style, and this is intended to aim to make quantum computing more accessible to software developers by automating the deployment process.

ACKNOWLEDGEMENTS

This work has been partially funded by MCIN/AEI/10.13039/501100011033 and by the European Union "Next GenerationEU /PRTR", by the Ministry of Science, Innovation and Universities (projects PID2021-1240454OB-C31, TED2021-130913B-I00, PDC2022-133465-I00). It is also supported by the QSALUD project (EXP 00135977 / MIG-20201059) in the lines of action of the Center for the Development of Industrial Technology (CDTI); by the Ministry of Economic Affairs and Digital Transformation of the Spanish Government through the Quantum ENIA project call – Quantum Spain project; by the European Union through the Recovery, Transformation, and Resilience Plan – NextGenerationEU within the framework of the Digital Spain 2025 Agenda; and by the Regional Ministry of Economy, Science and Digital Agenda of the Regional Government of Extremadura (GR21133).

ORCID

Javier Romero-Álvarez https://orcid.org/0000-0002-3162-1446
Jaime Alvarado-Valiente https://orcid.org/0000-0003-0140-7788
Enrique Moguel https://orcid.org/0000-0002-4096-1282
Jose García-Alonso https://orcid.org/0000-0002-6819-0299
Juan M. Murillo https://orcid.org/0000-0003-4961-4030

References

1. Aaronson S. The limits of quantum computers. *Scientific American* 2008; 298: 62-69. https://www.scientificamerican.com/article/the-limits-of-quantum-computers/.

2. Zhao J. Quantum Software Engineering: Landscapes and Horizons. *Arxiv.* 2020. doi: 10.48550/arXiv.2007.07047

3. Ravi GS, Smith KN, Gokhale P, Chong FT. Quantum Computing in the Cloud: Analyzing job and machine characteristics. *International Symposium on Workload Characterization (IISWC)* 2021: 39–50. doi: 10.1109/IISWC53511.2021.00015

4. Rahaman M, Islam MM. A review on progress and problems of quantum computing as a service (QcaaS) in the perspective of cloud computing. *Global Journal of Computer Science and Technology* 2015. https://globaljournals.org/item/5374-a-review-on-progress-and-problems-of-quantum-computing-as-a-service-qcaas-in-the-perspective-of-cloud-computing.

5. Piattini M, Serrano M, Perez-Castillo R, Petersen G, Hevia JL. Toward a quantum software engineering. *IT Professional* 2021; 23(1): 62–66. doi: 10.1109/MITP.2020.3019522

6. Hevia JL, Peterssen G, Piattini M. QuantumPath: A quantum software development platform. *Software: Practice and Experience* 2022; 52(6): 1517–1530. doi: 10.1002/spe.3064

7. Gonzalez C. Cloud based qc with amazon braket. *Digitale Welt* 2021; 5: 14–17. doi: 10.1007/s42354-021-0330-z

8. Leymann F, Barzen J, Falkenthal M, Vietz D, Weder B, Wild K. Quantum in the cloud: application potentials and research opportunities. *International Conference on Cloud Computing and Services Science* 2020. doi: 10.5220/0009819800090024

9. Kumara I, Heuvel WJVD, Tamburri DA. QSOC: Quantum service-oriented computing. *Symposium and Summer School on Service-Oriented Computing* 2021: 52–63. doi: 10.1007/978-3-030-87568-8_3

10. Papazoglou MP, Georgakopoulos D. Introduction: Service-oriented computing. *Communications of the ACM* 2003; 46(10): 25–28. doi: 10.1145/944217.944233

11. Moguel E, Rojo J, Valencia D, Berrocal J, Garcia-Alonso J, Murillo JM. Quantum service-oriented computing: current landscape and challenges. *Software Quality Journal* 2022: 1-20. doi: 10.1007/S11219-022-09589-Y

12. Alvarado-Valiente J, Romero-Álvarez J, Moguel E, García-Alonso J, Murillo JM. Technological diversity of quantum computing providers: a comparative study and a proposal for API Gateway integration. *Software Quality Journal* 2023: 1–21. doi: 10.1007/s11219-023-09633-5

13. Weder B, Barzen J, Leymann F, Zimmermann M. Hybrid Quantum Applications Need Two Orchestrations in Superposition: A Software Architecture Perspective. *International Conference on Web Services (ICWS)* 2021: 1–13. doi: 10.1109/ICWS53863.2021.00015

14. Garcia-Alonso J, Rojo J, Valencia D, Moguel E, Berrocal J, Murillo JM. Quantum Software as a Service through a Quantum API Gateway. *IEEE Internet Computing* 2021. doi: 10.1109/MIC.2021.3132688

15. Valencia D, Garcia-Alonso J, Rojo J, Moguel E, Berrocal J, Murillo JM. Hybrid classical-quantum software services systems: Exploration of the rough edges. *International Conference on the Quality of Information and Communications Technology* 2021: 225–238. doi: 10.1007/978-3-030-85347-1_17

16. Alvarado-Valiente J, Romero-Álvarez J, Garcia-Alonso J, Murillo JM. A Guide for Quantum Web Services Deployment. *International Conference on Web Engineering* 2022: 493–496. doi: 10.1007/978-3-031-09917-5_42

17. Johnson B. Qiskit runtime, a quantum-classical execution platform for cloud-accessible quantum computers. *Bulletin of the American Physical Society* 2022. https://ui.adsabs.harvard.edu/abs/2022APS..MART28002J.

18. Lubinski T, Granade C, Anderson A, et al. Advancing hybrid quantum–classical computation with real-time execution. *Frontiers in Physics* 2022; 10: 940293. doi: 10.3389/fphy.2022.940293

19. Hilton J. Building the quantum workforce of the future. *Forbes Technology Council* 2019. https://www.forbes.com/sites/forbestechcouncil/2019/06/19/building-the-quantum-workforce-of-the-future/.

20. Jabbari R, Ali bN, Petersen K, Tanveer B. What is DevOps? A systematic mapping study on definitions and practices. *Proceedings of the scientific workshop proceedings of XP2016* 2016: 1–11. doi: 10.1145/2962695.2962707

21. Gheorghe-Pop ID, Tcholtchev N, Ritter T, Hauswirth M. Quantum DevOps: Towards Reliable and Applicable NISQ Quantum Computing. *2020 IEEE Globecom Workshops* 2020: 1-6. doi: 10.1109/GCWkshps50303.2020.9367411

22. Weder B, Barzen J, Leymann F, Salm M, Vietz D. The quantum software lifecycle. *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software* 2020: 2–9. doi: 10.1145/3412451.3428497

23. Shahin M, Babar MA, Zhu L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE access* 2017; 5: 3909–3943. doi: 10.1109/ACCESS.2017.2685629

24. Schwichtenberg S, Gerth C, Engels G. From Open API to semantic specifications and code adapters. *International Conference on Web Services (ICWS)* 2017: 484–491. doi: 10.1109/ICWS.2017.56

25. Laso S, Berrocal J, García-Alonso J, Canal C, Manuel Murillo J. Human microservices: A framework for turning humans into service providers. *Software: Practice and Experience* 2021; 51(9): 1910–1935. doi: 10.1002/spe.2976

26. Grover LK. A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* 1996: 212–219. doi: 10.1145/237814.237866

27. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 1999; 41(2): 303–332. doi: 10.1137/S0097539795293172

28. Simon DR. On the power of quantum computation. *SIAM journal on computing* 1997; 26(5): 1474–1483. doi: 10.1137/S0097539796298637

29. Farhi E, Goldstone J, Gutmann S, Zhou L. The quantum approximate optimization algorithm and the sherrington-kirkpatrick model at infinite size. *Quantum* 2022; 6: 759. doi: 10.22331/q-2022-07-07-759

APPENDIX A. EXECUTING THE DEPLOYED QUANTUM SERVICES

This appendix describes the process for executing the deployed quantum services, which are web-based endpoints that allow users to interact with quantum computers.

Once the services containing the quantum algorithms have been generated and deployed, developers can access them through the web interface where they are hosted—via the URL returned by the continuous deployment pipeline.

When developers access the web interface, they will see an interface similar to the one shown in Figure 16, where the endpoints of the quantum and classical services—if these have been defined in the YAML of the specification—are located.

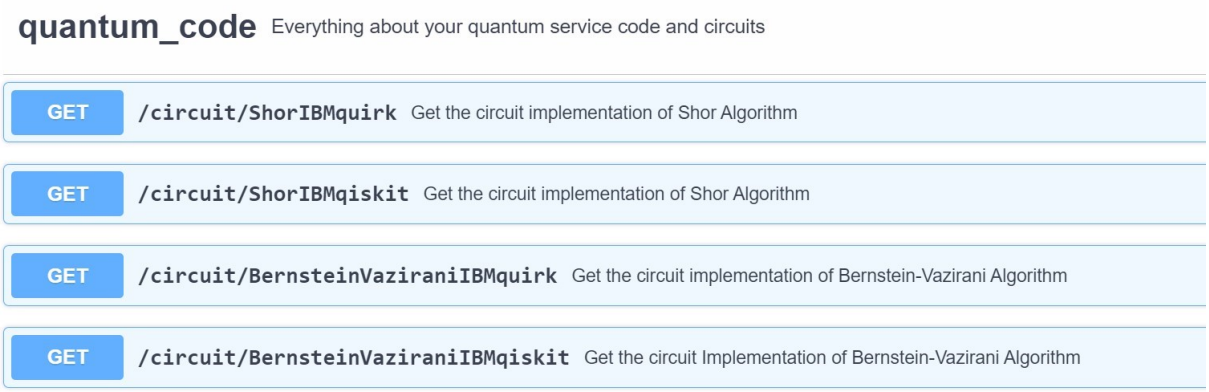


FIGURE 16 Examples of quantum services endpoints

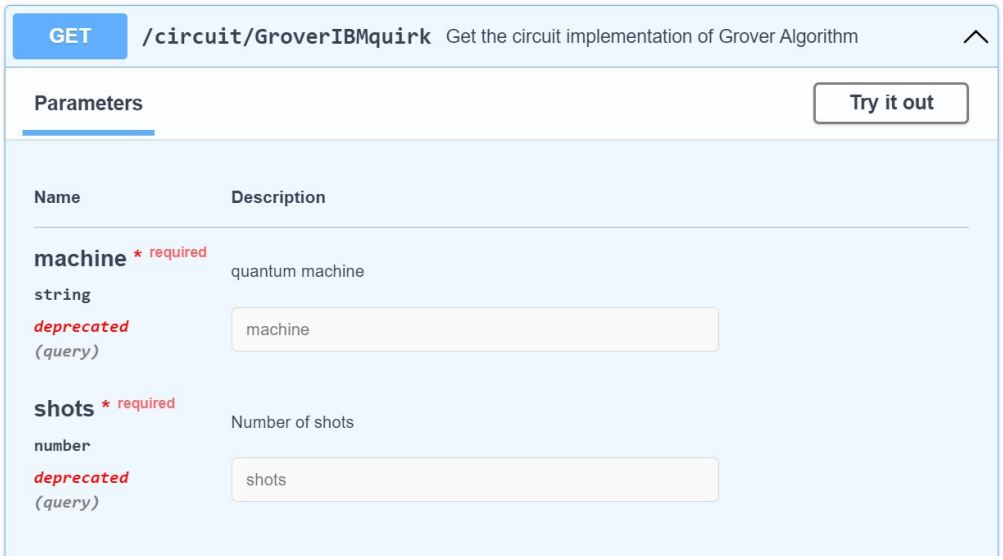


FIGURE 17 Parameters to execute a service

For the execution of these services, the developer must select one of them and click on the *Try it Out* option. With this, the parameters to perform the execution can be edited, as shown in Figure 17.

Specifically, in the *Parameters* section, the developer must indicate the **machine or simulator** where the execution is to be performed and the **number of shots**, to indicate the number of times the circuit is to be executed on the chosen quantum machine.

The only aspect the developer has to take into account is which service provider the selected service is developed for—AWS or IBM—to indicate a machine that corresponds to the provider.

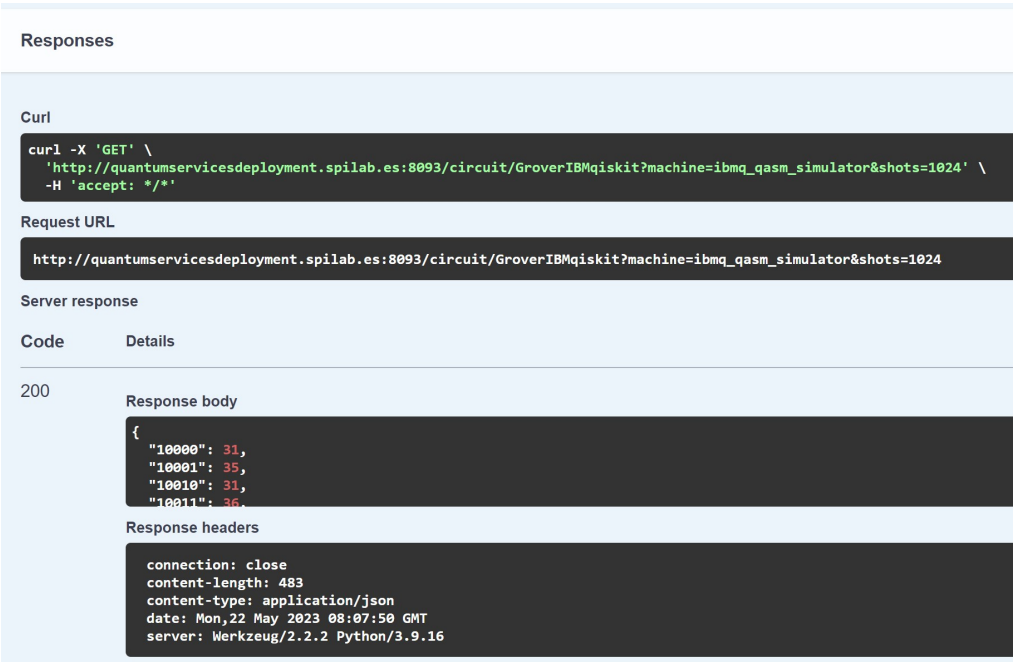


FIGURE 18 Result obtained from the execution of a service

After clicking on the *Execute* button, it will start the execution of the circuit on the selected machine and output a JSON with the results in the response body—Figure 18. These results represent the frequencies of occurrence of different quantum states after the execution of a circuit on a quantum machine.

Each key in the result corresponds to a binary state, where each digit represents the state of a qubit. The value associated with each key indicates the number of times that specific state was observed during the execution of the circuit. In the example, the state "10001" was observed 35 times, while the state "11101" was observed 19 times. These results provide information about the probability distribution of the quantum states in the experiment performed.

This roadmap has shown how to perform the execution of the deployed quantum services, although developers must verify the expected behavior and analyze the probability distribution obtained after the executions. However, with this tool, they save time and effort, as it is a simple process. In addition, developers are abstracted from the low-level details of quantum hardware and programming languages. This allows them to focus on application development without the need for in-depth knowledge of these technical aspects.

APPENDIX B. SURVEY

This appendix presents in detail the survey and the results obtained to confirm the questions raised and verify the usefulness of the pipeline presented.

Questions

The first part of the survey includes questions to obtain a profile of the participants. To do this, we posed the following general questions:

GQ1. What is your current employment situation? This question provides information to understand the situation of the participants. The options are:

a. Student
b. Researcher
c. Developer
d. Other

GQ2. What is your academic background? This question helps to understand the educational profile of the participants. The options are:

a. Software Engineer/Developer
b. Computer Science
c. Physics
d. Other

GQ3. What is your age range? The answer to this question provides an idea of the age distribution within the surveyed sample and helps to better understand the audience. The options are:

a. 17–24
b. 25–34
c. 35–44
d. 45–54
e. 55+

GQ4. How many years of experience do you have in Software Development? This question allows us to determine the total amount of time a person has spent on software development in his or her professional career. This question helps to understand the level of experience and expertise the respondent has in that specific field. The options are:

a. None
b. 0-4
c. 5-10

d. +10

GQ5. How many years of experience do you have in quantum computing? The answer to this question provides information about respondents' prior knowledge and experience in the field of quantum computing, which may be relevant in order to assess their understanding and perspective. The options are:

a. None
b. 0-2
c. 3-4
d. 5+

Concerning the second part of the survey, research questions were asked in order to analyze the current perception of quantum software development:

RQ1. In your opinion, what is the main aspect of Quantum Computing that should be improved from a software engineering point of view? This question was raised to obtain the perspective and opinions of respondents on specific areas for improvement at the intersection of quantum computing and software engineering. It is a free text question.

RQ2. Have you designed any quantum circuits and executed them in a quantum machine or simulator before? This question aims to evaluate the level of knowledge and practical experience of the respondents in the field of quantum computing. Specifically in the design and implementation of quantum circuits. The options are:

a. Yes, I have designed some quantum circuits and executed them in a quantum machine or simulator.
b. Yes, but only I designed the same circuits, I did not execute them.
c. No

RQ3. Have you developed or deployed any quantum services before? This question is related to the previous one, but in this case, the answers to this question will provide information on the familiarity of the respondents with the development of applications and services in the field of quantum computing. The options are:

a. Yes, I have developed and deployed one or more quantum services before.
b. No, but I have experience with quantum computing. I have worked with quantum computing before, but I have not developed or deployed any quantum services yet.
c. No, and I have no experience with quantum computing. I am not familiar with quantum computing, and I have not developed or deployed any quantum services before.

RQ4. How challenging do you find the process of generating and deploying quantum services using current methods? This question seeks to assess the degree of complexity and difficulties perceived by respondents in the development and implementation of quantum services using existing methods seen in the course. The options are:

a. Very challenging
b. Somewhat challenging
c. Not challenging at all
d. Not sure

RQ5. Is it essential for you to have tools that will allow the development and deployment of quantum services? This question is useful to know if the respondents see the need for tools that allow in terms of facilitating and facilitating the development process of quantum services. The options are:

a. Yes
b. No

After introducing the basic concepts of quantum computing and showing the structure of a quantum program with current development techniques, the pipeline for the generation and deployment of quantum services by adapting techniques from the DevOps methodology for continuous software integration was presented, with real examples of use. The objective was to corroborate the ease and usability of the process presented. To this end, the following questions were asked to check its usefulness:

RQ6. Have you understood the concepts explained in the course? This question allows us to know if the contents explained during the course have been understood or not, with a view to future improvements both in the proposed process and in the course. The options are:

a. Yes
b. No

RQ7. How much easier do you think it would be to develop and deploy quantum services using the semi-automatic method presented compared to the current methods? This question allows us to evaluate the opinion of the respondents on the improvement degree that the presented semi-automatic approach can offer in terms of simplicity and efficiency in the development and deployment of quantum services. The options are:

a. Much easier
b. Somewhat easier
c. Not much easier
d. Not sure

RQ8. How effective do you think the Continuous Deployment (CD) approach would be in simplifying and facilitating the development process of quantum services? Responses to this question will provide information on the respondents' expectations and perceptions of the potential benefits and advantages of the Continuous Deployment approach compared to existing methods. The options are:

a. Very effective. It would significantly simplify and facilitate the development process of quantum services.
b. Somewhat effective. There may still be some challenges or limitations that need to be addressed.
c. Not effective. Other methods may be more suitable for this purpose.
d. Not sure

RQ9. How important is it for you to have a simplified workflow for generating and deploying quantum services? This question allows us to determine the importance that respondents assign to having a simplified workflow for generating and deploying quantum services. The options are:

a. Very important. I think having a simplified workflow for generating and deploying quantum services is crucial to make quantum computing more accessible and practical for developers.
b. Somewhat important. I think having a simplified workflow for generating and deploying quantum services would be helpful, but it may not be a top priority.
c. Not important at all. The current process for developing and deploying quantum services may be complex, but it is necessary given the complexity and novelty of quantum computing.

Results

In the following, the results of the general questions GQ1, GQ2, GQ3, GQ4, and GQ5 are detailed:

GQ1. What is your current employment situation? The results obtained for this question are shown in Figure 7a. We can observe that almost all the participants (85%) are currently students and the rest are mainly researchers (8%).

GQ2. What is your academic background? The results obtained for this question are shown in Figure 7b. We can see that 51% come from the computer science field, 26% are software developers and 21% come from other academic fields—such as telecommunications or web engineering among others.

GQ3. What is your age range? The vast majority are between 25-34 years of age (62%), and the rest of the ages are mainly between 17-24 (15%) and 35-44 (15%).

GQ4. How many years of experience do you have in Software Development? The responses to this question vary as follows: 33% of participants have 0-4 years of experience, 31% have 5-10 years of experience, and the remaining participants have over 10 years of experience.

GQ5. How many years of experience do you have in quantum computing? Most of the participants do not have any experience in quantum computing (97%). This is because it is a relatively new field of research and because of the areas in which the course has been carried out.

The following results correspond to the questions RQ1, RQ2, RQ3, RQ4, and RQ5 that allow us to analyze the current perception of quantum software development:

RQ1. In your opinion, what is the main aspect of Quantum Computing that should be improved from a software engineering point of view? The general conclusion drawn from the answers is that the main aspect of quantum computing that should be improved from a software engineering perspective is the developer experience. Therefore, some key areas for improvement were highlighted by the participants, such as developing more accessible programming languages and tools for quantum programming; improving the availability and accessibility of quantum machines; standardizing access to quantum computing across different vendors to address the "vendor lock-in" problem; and increasing the level of abstraction in quantum computing. These are very useful responses as they reflect the same problems we are trying to address with our proposal.

RQ2. Have you designed any quantum circuits and executed them in a quantum machine or simulator before? The results of this question can be seen in Figure 8. Most of the participants (87%) have designed a quantum circuit but have not executed it on real machines.

RQ3. Have you developed or deployed any quantum services before? The results of this question can be seen in Figure 9. 97% of the participants have not developed or deployed any quantum services before but have worked in quantum computing before.

RQ4. How challenging do you find the process of generating and deploying quantum services using current methods? The results of this question can be seen in Figure 10. A significant percentage of respondents (38%) indicated that they find it to be somewhat challenging, while a slightly higher percentage (44%) described it as very challenging. This suggests that there are considerable difficulties associated with the current methods employed in the generation and deployment of quantum services.

RQ5. Is it essential for you to have tools that will allow the development and deployment of quantum services? The results of this question can be seen in Figure 11. Based on the results, it is evident that a majority of respondents (almost 80%) consider it essential to have tools that enable the development and deployment of quantum services. This is related to question RQ4 where also a high percentage is considered very challenging in the current process of development and deployment of quantum services.

The subsequent findings present the results pertaining to the questions that specifically assess the utility of the proposed process:

RQ6. Have you understood the concepts explained in the course? The results of this question can be seen in Figure 12. It is clear that the majority of participants (97%) have understood the concepts explained in the course. This high percentage of understanding indicates that the course content has been effectively communicated and that the proposed process is easy to understand for developers.

RQ7. How much easier do you think it would be to develop and deploy quantum services using the semi-automatic method presented compared to the current methods? The results of this question can be seen in Figure 13. Based on the responses, it can be inferred that there is a range of opinions (64%) regarding the ease of developing and deploying quantum services using semi-automatic methods compared to current methods. Additionally, 26% of the respondents believe that semi-automatic methods would not make the process "*much easier*" and 10% of the respondents are unsure about the impact of semi-automatic methods on the ease of development and deployment of quantum services.

RQ8. How effective do you think the Continuous Deployment (CD) approach would be in simplifying and facilitating the development process of quantum services? The results of this question can be seen in Figure 14. The majority of respondents (59%) considered a CD approach to be "*very effective*" in significantly speeding up and facilitating the quantum services development process and 23% of the respondents consider a CD approach to be "*somewhat effective*", indicating that while it has advantages, there may still be challenges or limitations that need to be addressed. The remaining percentage is not sure or does not consider it effective.

RQ9. How important is it for you to have a simplified workflow for generating and deploying quantum services? The results of this question can be seen in Figure 15. A majority of participants (69%) consider it very important to have a simplified workflow for generating and deploying quantum services. (26%) of the respondents express that having a simplified workflow for generating and deploying quantum services is somewhat important. On the other hand, only 5% of the respondents consider it not important at all.

How to cite this article: Javier Romero-Álvarez, Jaime Alvarado-Valiente, Enrique Moguel, Jose García-Alonso, and Juan M. Murillo (2023), Enabling Continuous Deployment Techniques for Quantum Services, *Softw Pract Exper*