Received: 24 May 2021 | Revised: 15 November 2021 | Accepted: 9 December 2021

DOI: 10.1002/spe.3064

EXPERIENCE REPORT

WILEY

# QuantumPath: A quantum software development platform

**Jose Luis Hevia**[1] | **Guido Peterssen**[1] | **Mario Piattini**[1,2]

[1]aQuantum, Madrid, Spain

[2]aQuantum, Alarcos Research Group, ITSI (Institute of Information Technologies & Systems), Universidad de Castilla-La Mancha, Ciudad Real, Spain

**Correspondence**
Mario Piattini, aQuantum, Alarcos Research Group, ITSI (Institute of Information Technologies & Systems), Universidad de Castilla-La Mancha, Ciudad Real, Spain.
Email: mario.piattini@uclm.es

**Funding information**
European Regional Development Fund; Centre for the Development of Industrial Technology of the Ministry of Science and Innovation of Spain

**Abstract**

Quantum computing has experienced a breakthrough. Several companies are taking up the challenge of designing and manufacturing quantum computers, and the supply of tools for quantum software development is growing all the time. This article addresses quantum software development toolkits and introduces the 'QuantumPath' platform. In developing QuantumPath, our aim is to fulfil certain principles such as: agnosticism, extensibility, integration, independency, optimisation, scalability, security, usability and software engineering support. This article presents both the architecture itself as well as the main tools that compose QuantumPath, in order to illustrate the support which platform provides to the development and execution of quantum software.

**KEYWORDS**

quantum computing, quantum toolkits, QuantumPath

## 1 | INTRODUCTION

As Lloyd states: 'The history of the universe is, in effect, a huge and continuous quantum computation. The universe is a quantum computer'.[1] Quantum computing is based on exploiting the properties of quantum mechanics, which is the field of physics that describes the behaviour of nature at subatomic levels, and is the deepest explanation known to science so far.[2]

Quantum computers, which can be defined as 'computing devices that store[s] information in objects called qubits and transform[s] it by exploiting certain very special properties of quantum mechanics',[3] are already in use. Within the label of quantum computers, we can also find quantum simulators. In these machines, when implementing the quantum algorithm, it is necessary to compile the circuits in accordance with the topology of the classical chip used for execution; accordingly, the quantum algorithm is simulated in the end on the classical hardware. And there are also true quantum computers, based on the new technologies of trapped ions, superconductors, photons, and so forth,[4] and among which are analogue and digital quantum computers.

In analogue quantum computers – also called 'quantum annealing computers' (of which D-Wave machines are an example) – the starting point is a system of qubits in an initial state and a system which is operated by manipulating the analogue values of the system's Hamiltonian. Hence, this type of quantum computer uses the adiabatic theorem to find a global optimum in a discrete optimisation problem by using a quantum fluctuation process. Note that annealing can also be used with non-quantum architectures; in fact, the term 'Ising machine systems' groups together quantum annealing, laser network-based and annealing systems.[5] The latter two are non-quantum architectures. Examples of computers based on annealing on digital circuits (digital annealer) are those of Fujistsu.

**Abbreviations:** BQP, bounded-error quantum polynomial time; GPU, graphics processing unit; NISQ, noisy intermediate-scale quantum; QASM, quantum assembly language; QEC, quantum error correction; QPU, quantum processing unit; SDK, software development kit; UI, user interface.

Digital or gate-based quantum computers can be considered as being the generalisation of classical computing, in which bits are replaced by qubits. It is precisely in the qubits where information is encoded, breaking down a problem into a set of basic operations performed by gates that obtain a 'digital' result. Google, IBM, Rigetti, IonQ and Honeywell computers are all gate-based. These types of computers are more sensitive to interference from their environment, so, in terms of reliability, currently we only have what is known as 'noisy intermediate-scale quantum (NISQ)'.[6]

Most quantum computers today present a hybrid architecture in which a classical computer acts as master, controlling the quantum computing processes and sending the appropriate instructions to the quantum computer (slave), taking into account aspects such as the execution times of the quantum gates or the coherence times of the qubits. Therefore, a quantum processing unit (QPU) can be seen as analogous to what currently happens with the graphics processing unit (GPU), which is a dedicated co-processor used to accelerate certain mathematical operations and image processing.

All these quantum systems share the characteristic that not only do they allow us to simulate nature much better, but they can also execute algorithms that require massive parallel computations, which for 'classical' computers would be impractical. This is due to their being able to approach a new class of problems called 'bounded-error quantum polynomial time' (BQP).[7] As a result, significant investments in quantum systems are currently taking place in the fields of security, artificial intelligence, communications, optimisation, pharmacology, medicine, chemistry, materials development and so forth.

On the software side we are confident that 'quantum computing may become the main driver of a new "golden age" of software engineering',[8] in the same way as occurred with other advances such as structured programming, object-oriented or DevOps.[9] With the great advantages that quantum computing offers us, the opportunity exists to benefit from the same leap forward as the pioneers of software engineering in the 60s of the last century experienced.[10]

Of course, all these advantages will only be realised if we are able to develop a true engineering for quantum software[11–13] and to create the right tools, not only in terms of quantum programming languages but also quantum software development environments.

In this article, we introduce QuantumPath, which is a quantum software development platform to support the design, implementation and execution of quantum software applications.

The rest of this article is organised as follows: Section 2 introduces the most important existing quantum software development toolkits; Section 3 provides an overview of the QuantumPath platform principles, functionalities and architecture; Section 4 offers an example of QuantumPath use; and Section 7 present the conclusions and future work.

## 2 | RELATED WORK

In this section, we review the state of the art regarding the main quantum software development toolkits.

### 2.1 | Quantum computer vendors toolkits

Quantum computer manufacturers provide both local simulators and cloud resources to access real quantum machines. The specific language of the quantum computer runs on the particular QPU for which it is designed. On top of this assembly language, high-level and general-purpose languages (usually Python) are also provided to compose the circuit (in gate-based quantum computers) or the solver (in quantum annealing computers) and to send it to the execution services. Each manufacturer provides its own rules of access to the environment and its own versions of approved languages. In addition, specific libraries are frequently offered as a set of extensions to the programming language.

The main quantum computer toolkits vendors are:

- D-Wave.[14] Ocean is the software development kit (SDK) developed by D-Wave, which encompasses a set of tools such as compilers, full-stack libraries, samplers, solvers and so forth, that allow modelling optimisation problems to be run on their simulators and quantum machines.
- Fujitsu[15] has launched Quantum-Inspired, a set of services, tutorials and Python libraries that facilitate work on designing and solving complex optimisation problems using the digital annealing technique.
- IBM.[16] Qiskit is an open-source Python SDK developed by IBM, which allows the design of applications based on quantum circuits. The Quantum Experience web platform provides a set of higher-level tools that facilitates the creation of

quantum applications and access to the company's simulators and quantum computers. IBM has also produced Quantum Composer, a graphical editor for quantum circuits which abstracts from programming in Qiskit, and Quantum Lab, an installation-free, cloud-based Jupyter notebook environment that allows one to code and run quantum circuits with Python and Qiskit.

- Microsoft.[17] The Microsoft Quantum Development kit is the development framework for Q#, a specific programming language for designing quantum circuits, integrated in the Visual Studio development environment. Azure Quantum is Microsoft's quantum platform in the cloud, which allows one to create and run Q# programs on simulators or on quantum computers from different vendors (IonQ and Honeywell are supported by default), in addition to formulating optimisation programs that can be solved directly on classic Azure hardware.
- Rigetti[18] offers a set of open-source tools through its Forest SDK, ranging from higher-level language interfaces, such as pyQuil, and Groove, to Quil (its assembly language); simulation environments such as QVM; and circuit optimisation and compilation software. Forest is an environment designed for experimenting with and executing stand-alone quantum algorithms.
- Xanadu provides two different products: Strawberry Fields,[19] a Python library for the design of quantum circuits oriented to solve problems based on graphs or networks, with specific functions for machine learning and the chemical industry field, and PennyLane,[20] a cross-platform Python library for learning quantum programming and optimising hybrid computations. PennyLane has add-ons to enable work with other quantum platforms, such as IBM's Qiskit.

## 2.2 | Third party toolkits

There are many other quantum software development environments,[21,22] and several cases of open-source ones. Some of these are being developed at universities, others by start-ups. In this section we will summarise the most important ones:

- Orquestra[23] has been developed by Zapata Computing, which permits working with the frameworks and hardware of the main quantum vendors through a set of modular tools based on workflows.
- ProjectQ[24] is an open-source software framework implemented in Python at ETH Zurich. It allows users to create their quantum programs in this language, and subsequently to translate them to any kind of back-end, be it a simulator running on a classical computer or a real quantum computer. It is currently integrated with the IBM Quantum Experience platform and work is underway to support other hardware platforms in the future.
- Quantum Inspire.[25] A quantum computing platform designed and built by QuTech (founded as a collaboration between Delft University of Technology and the Netherlands Organisation for Applied Scientific Research), its aim is to provide users with access to a variety of technologies with which to perform quantum computations. It has a variety of utilities to program quantum algorithms, execute these algorithms and examine the results, and includes a graphical interface for programming in quantum assembly language (QASM) and visualising operations on circuit diagrams.
- QuantumPath,[26] the tool we created and which we present in this article, was conceived as a Quantum Software Development and Lifecycle Application Platform. It is an ecosystem of tools, services, and processes that simplify the development of quantum algorithms into hybrid information systems.
- Quantum Programming Studio[27] is a web development environment for building gate-based quantum circuits through a drag-and-drop editor, or through the use of Open QASM code. It is a platform application that allows circuits to be exported to different programming languages and can be run on various simulators and quantum computers, including Rigetti Forest, IBM Qiskit, Google Cirq, Microsoft Quantum Development Kit, as well as on Amazon Braket platforms.
- Strangeworks Quantum Computing platform[28] is a web environment for the development of quantum applications, which supports many frameworks (Q#, QISkit, Forest, CIRQ, Ocean, etc.), thus allowing work with simulators and quantum computers from the main suppliers. It provides an automated Kubernetes system that provides the end user with the various server-deployed tools that are needed to work with a given technology.

An analysis of the main characteristics of the main quantum software development toolkits can be found in Reference 29.

# 3 | QUANTUMPATH

## 3.1 | Design principles

When creating QuantumPath we set out to adhere to the following principles:

- Agnosticism. The platform should be able to support both quantum gate-based and annealing technology.
- Extensibility. Since quantum technologies are in the process of continuous research and evolution, the system must be capable of adapting to change through a complete and well-designed model of extensions.
- Integration. The environment should enable the integration of quantum/classical (hybrid) information systems.
- Independency. The environment should enable programmers to be independent of the specific details of each platform and language, based on the principle of 'write once, run everywhere'. Therefore, it masks the complexities of the different environments by supporting the necessary transformations and automating the whole process through efficient tools.
- Optimisation. The environment needs to collect and analyse all the stored telemetry.
- Scalability. The environment must be scalable. It shall be deployed in as many servers as necessary to guarantee a growth according to the number of users and processes generated.
- Security. The system shall be designed to be secure. It shall provide secure access to the service layers, as well as guaranteeing the protection of the system assets.
- Software engineering. The platform shall support the quantum software life cycle and its engineering.
- Usability. The platform shall help to visually design the quantum assets of the application and to define the environment's requirements, and shall explore the results using a unified scheme, irrespective of the particular language of the quantum computers.

## 3.2 | Functionalities

The QuantumPath Core offers different functionalities:

1. Management of solutions and their assets, which includes all the elements necessary to compose a 'quantum application'. These include: (application) solution and its relationships to the quantum execution context in which it will unfold; quantum circuits and their different approaches, depending on the type of technology used; direct code units, when more direct contact with a particular machine is required; intermediate language treatment of the assets of a quantum application; and main flows for the coordination of the all necessary elements that make up an algorithm and which organise its execution control. All the data elements stored in the cloud system are stored encrypted from origin to storage, so as to comply at all levels with the privacy of knowledge premise of the product.
2. Tools for the design, construction, testing and execution of quantum assets from the context of both agnostic and platform-specific solutions: visual designers of gate-based circuits, the Annealer Compositor, and the direct code editor to solve lower-level needs.
3. Connection Points, which make possible the interconnection of quantum applications in the ecosystem of classic solutions. Using a clear publishing service, a concise layer of REST API services is provided such that any classical application can consume the quantum algorithms stored in the system with minimal effort. For example, the Connection Point (/api/connectionPoint/GetQuantumExecutionResponse/jobtoken/idSolution/idFlow) probes whether the job token job of the identified solution and application has been completed and what its outcome is:

```
HTTP GET HEADERS
  content-type: application/json
  Authorisation: Bearer tokenJWT
RESPONSE
{
"ExitCode": "OK",
"ExitMessage": null,
```

"ExecutionData": {
"Solution": "WEBINARAnneal",
"Flow": "BoxFlows",
"Device": "dwave_ExactSolver_simulator",
"Histogram": histogram struct",
"Duration": 157226789
}
}

4. Enterprise backend, which is responsible for the complete operation of the platform. A backend that – by design – contemplates security, high availability, load balancing and asynchronous customer processing. Fully scalable and reliable, it provides the necessary components for all work to be processed to the customer in a decoupled way and to allow for the launch of the execution units in the best possible context. This backend manages the approved connections of quantum simulators and quantum computers to suppliers; and will also collect all the telemetry from the process, thus providing knowledge and automatic assistance wherever needed.

QuantumPath allows the user to work with different quantum technologies such as those from IBM, Microsoft, Rigetti, D-Wave, Google and Fujitsu, as well as with third-party quantum computing simulators such as QuTECH or CTIC.[30] As shown in Figure 1, both the circuit editor and the annealing editor generate a proprietary metalanguage that is agnostic to each vendor's quantum technology. The corresponding metalanguage is then transpiled in design time to the programming language supported by each manufacturer and executed in the target environment. QuantumPath also provides different working pipelines that allow one to work out both fully agnostic and partially agnostic solutions or, if necessary, to do so directly on a vendor platform.

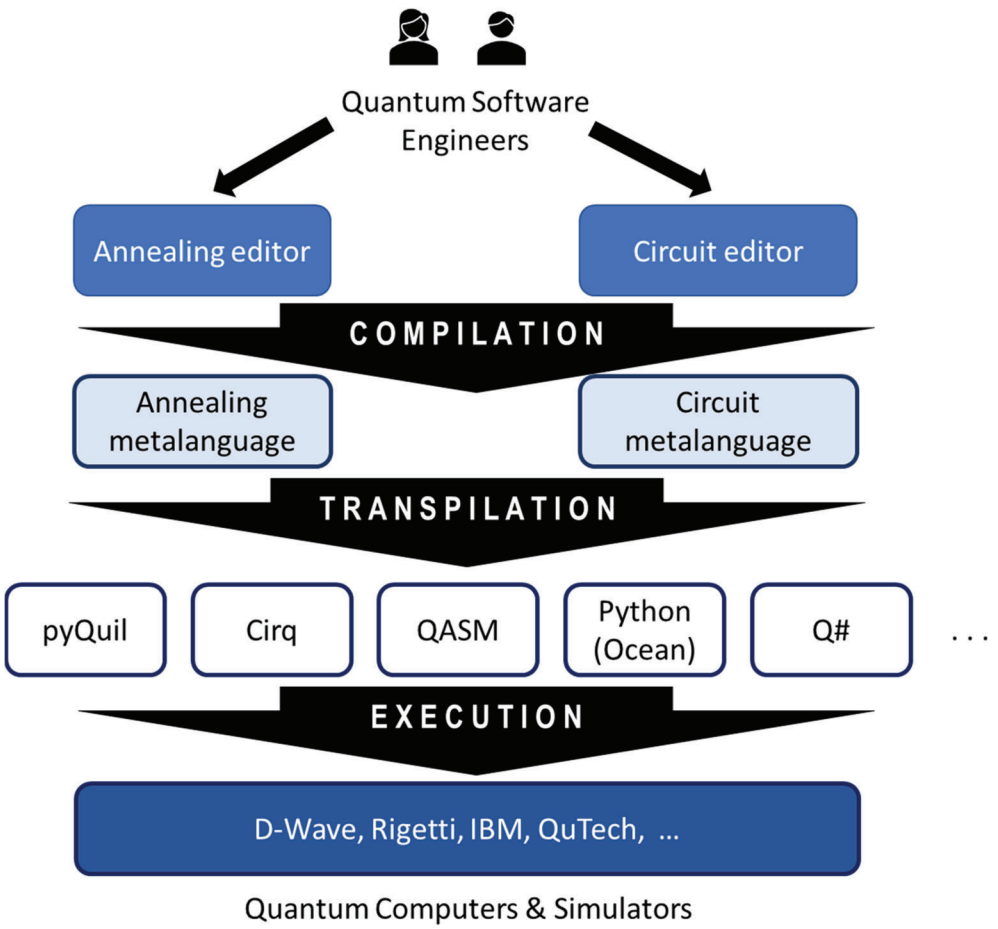


**FIGURE 1** General operation of QuantumPath

## 3.3 | Main design tools

QuantumPath allows quantum algorithms to be designed visually, using different tools. In Figure 2, these tools are represented in the framework of the other components of the QuantumPath platform.

### 3.3.1 | Circuit editor

The circuit editor permits the graphical design of quantum gate circuits (Figure 3) through a web user interface (UI) built upon the Quirk Quantum Circuit Simulator,* with drag and drop support to compose the circuit. As we can see in Figure 3, this version of the QuantumPath is highly customised and adds some new features with respect to the original Quirk version.

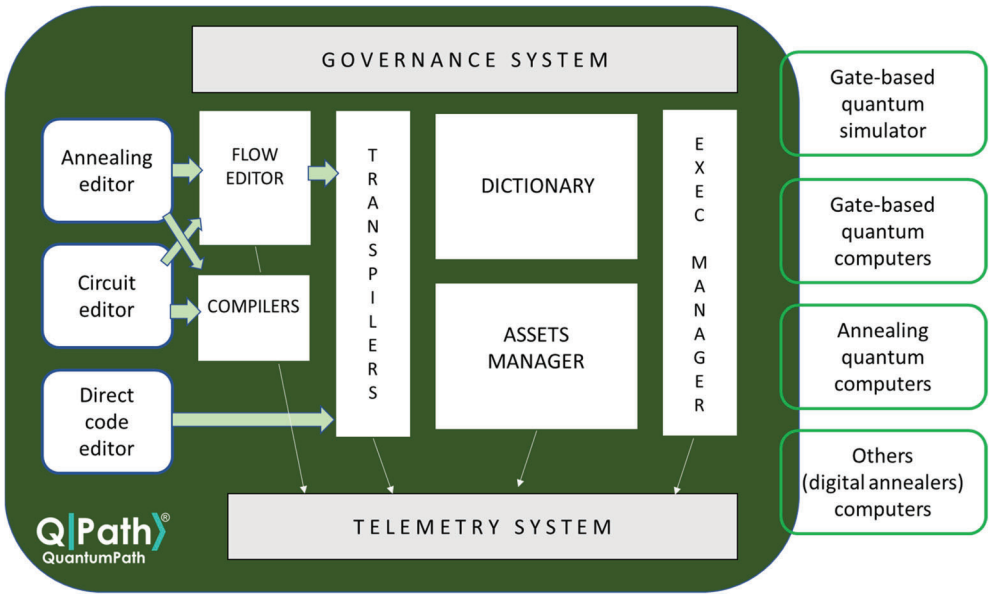


FIGURE 2 Overall view of the QuantumPath platform components

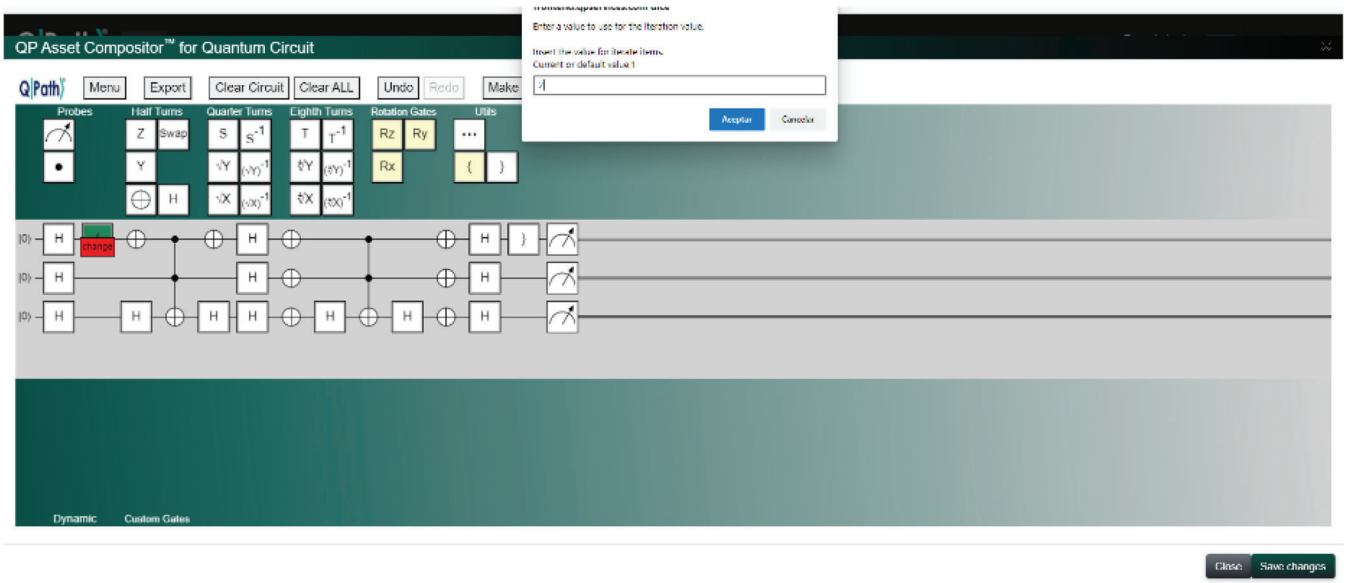


FIGURE 3 Circuit editor of QuantumPath

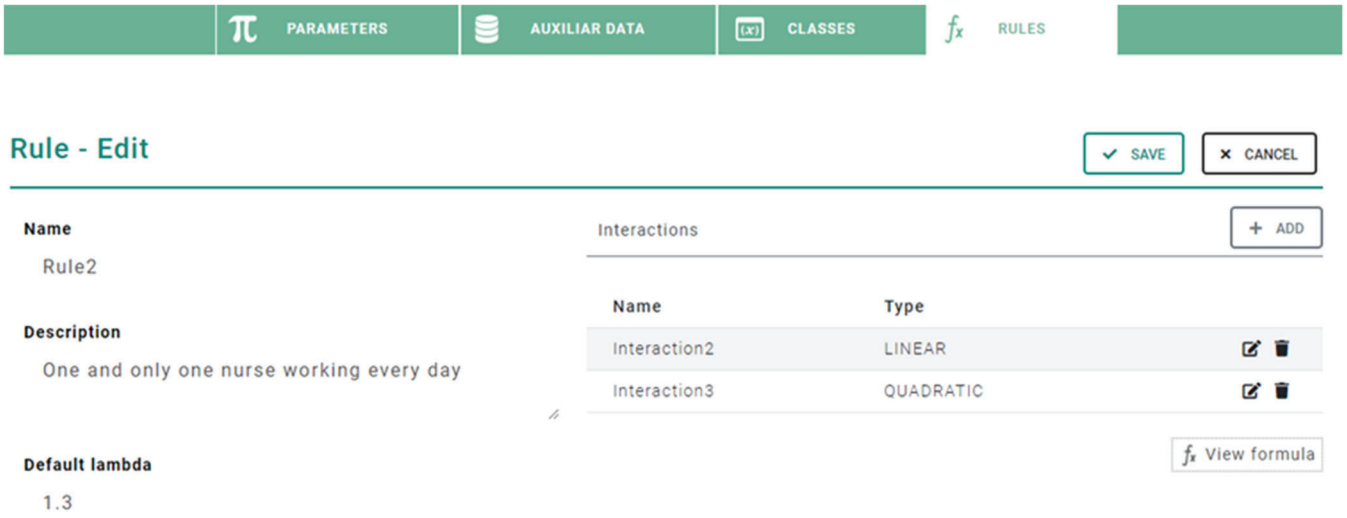


**FIGURE 4** Annealer compositor of QuantumPath

### 3.3.2 | Annealer compositor

The annealer compositor is a high-level tool that allows one to model optimisation problems in a fully graphical and visual way (Figure 4), without needing to use any programming language.

The design of the optimisation problem is simplified to the definition of four elements: parameters, auxiliary data, classes of variables, and rules. Once the editing is completed with the annealer compositor, the system compiles the circuit, and then generates the corresponding code in annealing metalanguage.

### 3.3.3 | Flow editor

As with any classical solution, we can take circuits as components that need to be orchestrated in an application where there can be multiple steps. So, to have an application in which to launch the circuits, we will need to define a flow using the editor shown. The flow has the responsibility for ensuring the coordination between the circuits and all the elements in which QuantumPath will be executed.

### 3.3.4 | Direct Code editor

It is also possible to use a Direct Code editor to edit the metalanguage code generated by all the above-mentioned tools. With the Direct Code feature a specialist in the quantum vendor's specific technologies can enter code in the vendor's own language (qASM in the case of QISKIT, Python OCEAN in the case of D-Wave Ocean, etc.). Two use cases can be examined: QuantumPath managed code – functions that isolate the specialist from the hardware and provider's libraries – or pure code from the manufacturer with all its requirements (in this case, QuantumPath behaves as a proxy of the provider's code).

## 4 | EXAMPLE OF OPERATION

In this simple example, we show the sequence of steps that must be followed to define and execute a quantum application.

We begin by defining a solution, which is a concept that allows us to abstract all the details of the particular SDKs which are to be deployed (QISKIT, Ocean, etc.), the language and the environment with which we are going to program (direct Python, Q#, Jupyter notebook, etc.), and the necessary files as well as the place where we will keep them (local disk, http repository, GitHub, etc.). The solution then has the responsibility of connecting the different available quantum

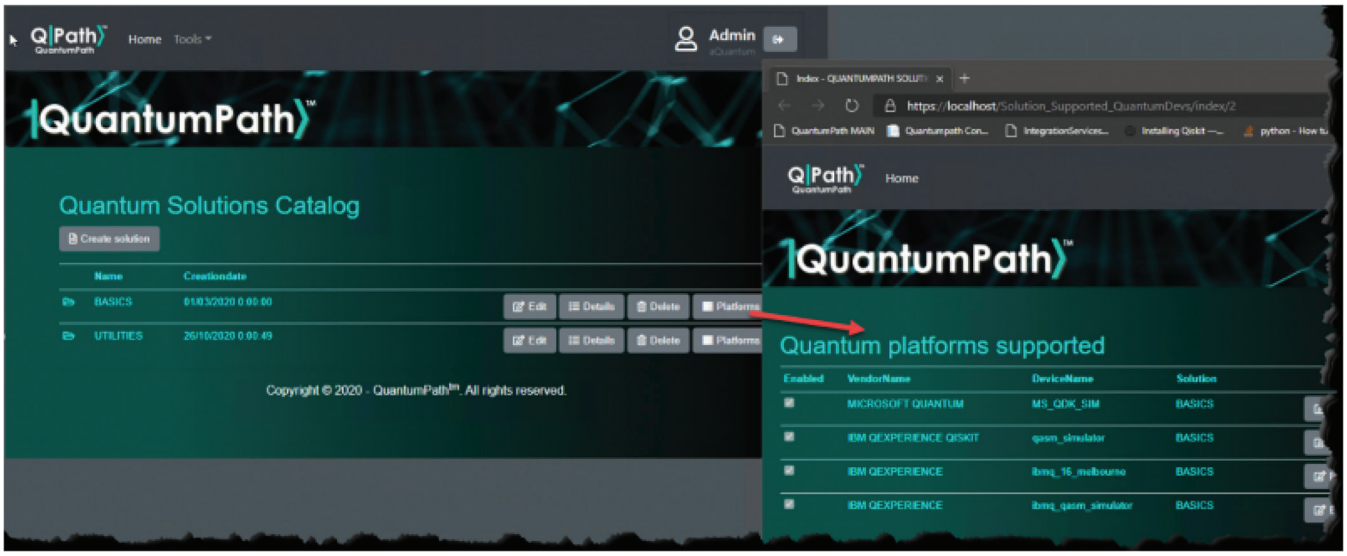


FIGURE 5 Solution management

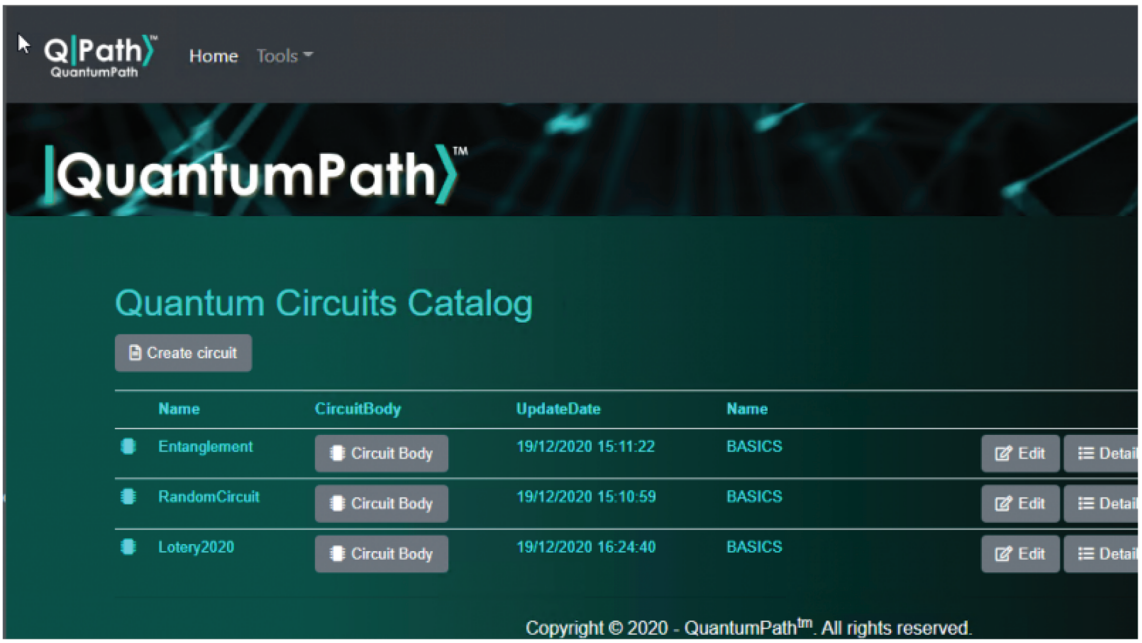


FIGURE 6 Quantum circuits catalogue

platforms (taking account here of their requirements, their different versions, and the relevant SDK) to the assets that we generate with QuantumPath. This allows us, among other things, to limit certain quantum technologies to the context of the problem and to licence them. In addition, from the beginning we can decide whether to use the path of quantum gate technologies or annealing technologies (see Figure 5).

Associated with the solution, we add, edit and manage the quantum assets with which we will solve a certain problem. So, for example, we may have a quantum circuit catalogue (Figure 6) in which we collect all the circuits defined using the circuit editor of QuantumPath (shown in Figure 3).

For each of the assets several metadata are stored (see Figure 7) which can be modified according to the engineer's needs when building the quantum software application.

As we have already mentioned, in order to have an application in which to launch the circuits we will first need to define a flow. In the case of our previous example of entanglement, a flow would be defined as is shown in Figure 8.

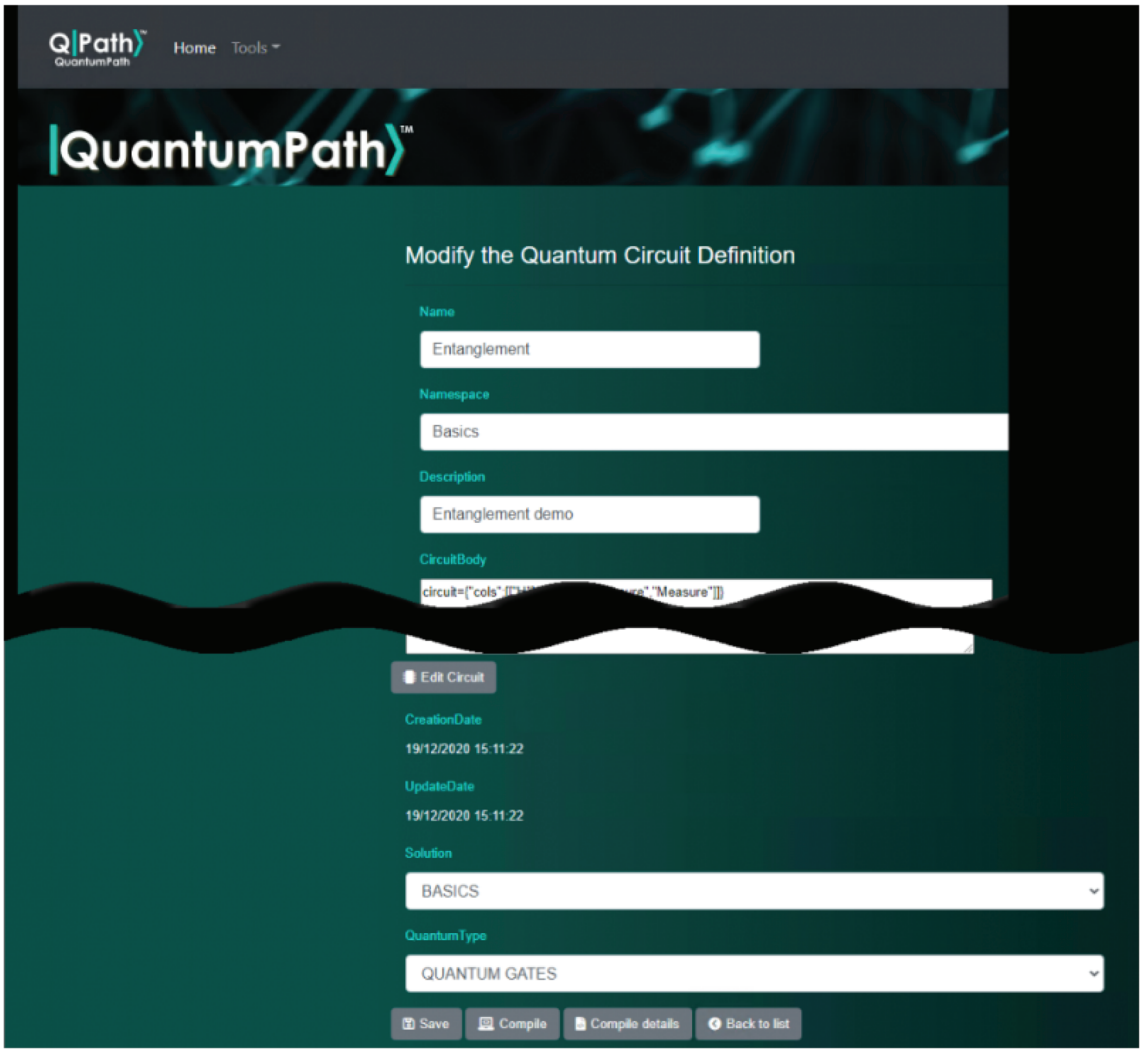


**FIGURE 7** Quantum circuit definition metadata

Once this work has been completed, we will be able to carry out the experiments. For this, QuantumPath provides the execution dashboard (Figure 9). Using this dashboard allows us to coordinate the necessary executions and check the results, it being possible, as mentioned previously, to select with great flexibility the execution destination from among those associated with the solution: that is, a real quantum machine, a quantum simulator, both, or all of those available.

The QuantumPath CORE, through its process manager, will be responsible for carrying out all the necessary actions to respond to the user's needs and to provide the different consolidated results. It is important to note at this point that all launching of experiments is totally asynchronous, and to keep in mind that, depending on the response capacity of a real quantum machine, the level of licence available, and the queue of requests in progress (which can be hundreds and thousands of experiments globally), it can take hours (or even days) to collect a response.

A more complete demonstration, walking the user through two examples of quantum-gates based circuit design and annealing-based optimisation, can be found in the aQuantum YouTube channel.[†]

QuantumPath's telemetry learning can help target selection spread the load by delivering results as quickly as possible and thus benefiting the overall environment. Its capability to allow the refinement of assets and to launch them in advance against local/distributed simulators also has the benefit of not overloading a real machine in continuous experimentation and can even positively influence the economics of projects in their development phase (so leaving those possibilities open until the end of the development). Additionally, the fact that the execution data is effortlessly stored in the system allows the results to be consulted later without the need to run the experiment again, so avoiding any overloading of the available machines.

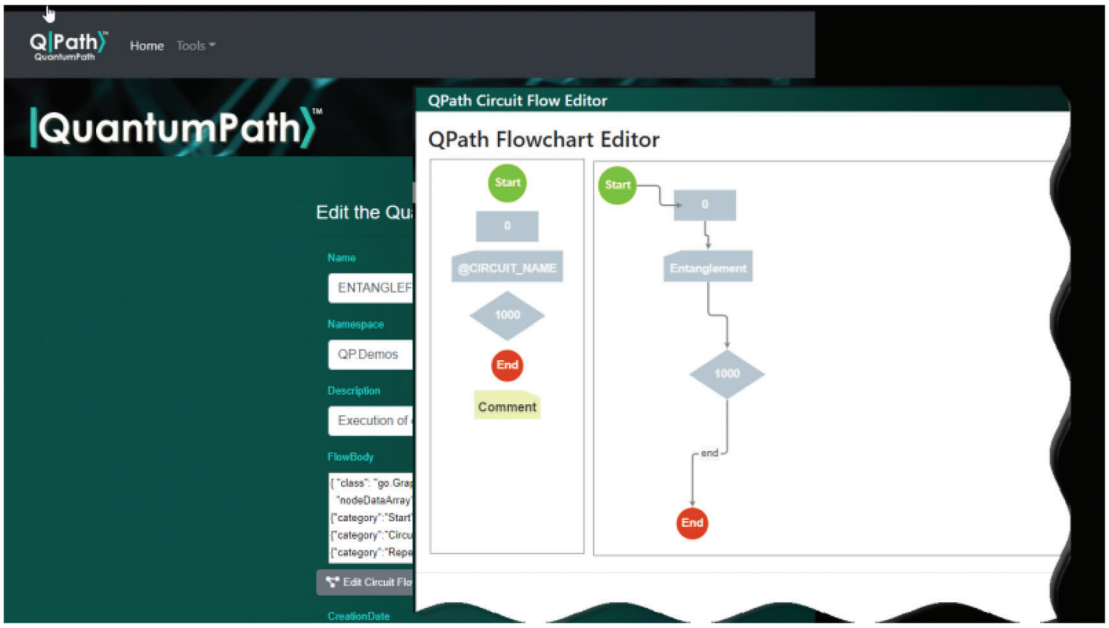


FIGURE 8 Editing and designing a flow

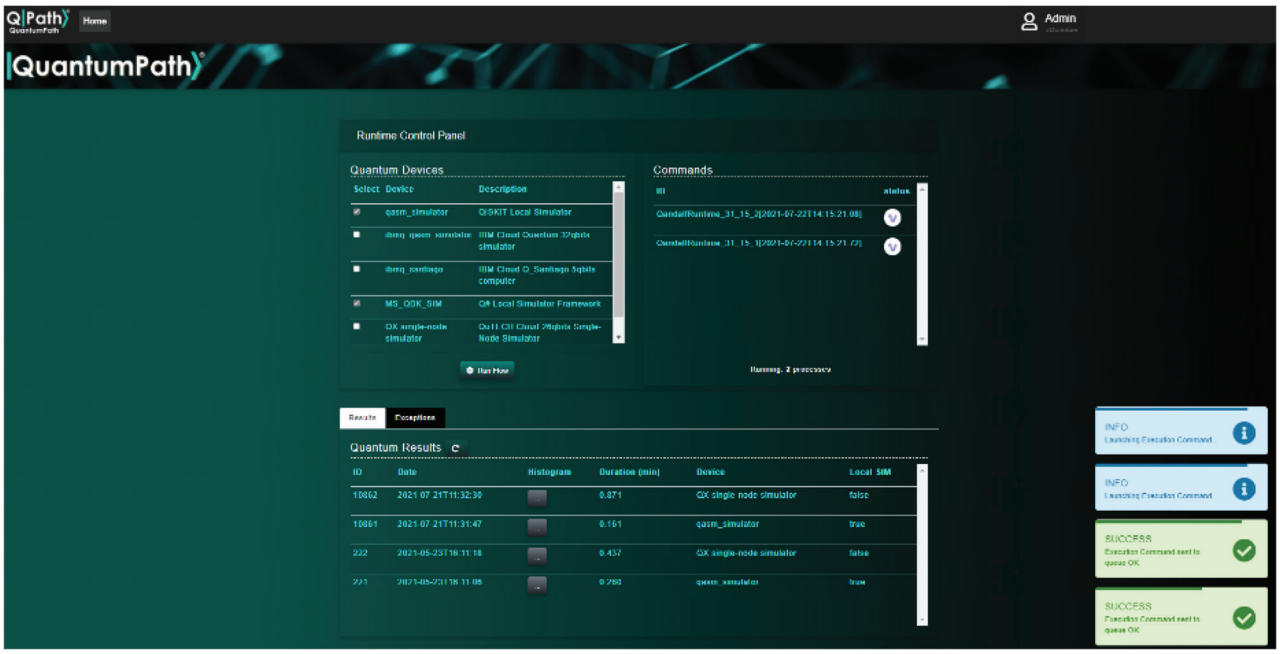


FIGURE 9 Execution dashboard

## 5 | BENEFITS OF QUANTUMPATH FOR QUANTUM COMPUTING

Thanks to the design principles used to guide the construction of QuantumPath (see Section 3.1), some important problems that often arise in quantum software development can be alleviated.

One of the main problems in quantum development is the lack of standardisation (due to the lack of maturity of the technology), which means that to use most of the tools requires learning a different language and environment for each quantum computer, which is inconvenient both in the learning curve and in having to redo all the work of specification and design of the quantum program every time the quantum computer is changed or even every time the manufacturer

**TABLE 1** Main quantum software development problems and QuantumPath's solutions

| Quantum software development problems | QuantumPath solutions |
|---|---|
| Lack of standardisation | Agnostic platform both for gated-based and annealing quantum computing |
| Steep learning curve | |
| Re-implementation | |
| Expensive testing | Integration of quantum simulators |
| Difficult design | High-level interfaces |
| Interaction in 'hybrid' systems | Connection Points |
| | qSOA |
| Deployment and execution optimisation | Telemetry System |

of the quantum computer updates its version. One great advantage of an agnostic platform such as QuantumPath is that it accelerates the adoption curve of quantum computing, allowing the development of algorithms independent of the quantum computer and its technology, and offers access to execution on different vendors without needing to change the source code.

Another aspect derived from the previously mentioned feature is the advantage offered by QuantumPath of testing the software that implements quantum algorithms in quantum simulators (which do not have the problems of inconsistency and errors of the current NISQs), thus allowing software engineers to concentrate on debugging and testing the quantum software (which is already a quite complex task as account has to be taken of aspects such as superposition or entanglement) and to isolate themselves from the problems caused by the quantum hardware. The platform also allows one to subsequently run – without any additional effort – the same program on the final quantum computer. This also represents a great economic saving due to the current costs of executing quantum computers.

A significant advantage in having the QuantumPath interfaces available when defining quantum algorithms is that software engineers and users can focus on high-level problem and solution definition and avoid over-complicated designs with aspects far removed from the domain logic. This was the case in the Fintech proof of concept that we carried out at CaixaBank, the leading financial group in retail banking in Spain.[31] In fact, this proof of concept demonstrated the ease of life cycle and deployment of quantum software, and the time savings from design to implementation when compared to classical software.

Of course, quantum systems will not replace existing systems but will instead have to coexist with them, so the aspect of integrating 'classical' and 'quantum' software parts in hybrid systems is fundamental in practice. For example, in the case of the QHealth ('Quantum pharmacogenomics applied to ageing') project,[32] the aim of which is to carry out optimisations and simulations on the correlation among the genetic and other variables related to the health history of patients, its execution in classical hardware is not possible in acceptable timescales. The quantum parts of the system must work in combination with classical health applications, giving its outputs to medical professionals involved in prescribing drugs to older adults. The Connection Points and the qSOA architecture[33] supported by QuantumPath greatly facilitate the interconnection of quantum applications with existing classical systems and the construction of efficient hybrid systems.

Finally, we would like to highlight another significant issue when deciding on which computer it is more convenient to run a quantum program, as they differ in efficiency, error tolerance, and cost, and much depends on the type of algorithm or circuit to be executed. For this purpose, as shown in Figure 2, QuantumPath collects in its telemetry system database a large amount of data on the different executions of the programs on different systems, which data is used for an AI-based system to decide or to propose to the engineer the most suitable platform in each case.

In Table 1, we summarise the main problems in quantum software development that would benefit from QuantumPath.

## 6 | LESSONS LEARNED AND NEW FEATURES TO ADDRESS

During the development of the QuantumPath tool we have learned several lessons, which can be summarised as follows:

LN1: Despite the lengthy experience (more than 30 years) that each of us has in developing tools in classical computing, we have to recognise that the learning curve of quantum computing is very pronounced. The paradigm shift from structured to object-oriented programming, for example, is not the same as the shift from classical to quantum computing in which the very basics of computing change.

LN2: Quantum software platforms and toolkits are difficult to use in industry. Assuming that the quantum software engineer knows how to incorporate each product to its corresponding platform, they do not bring much context support to the generation of quantum algorithms.

LN3: Several of the tools we have integrated into QuantumPath have been released in different versions (not always compatible with the previous ones), and some features have been discontinued – with the result that we have had to modify our platform.

LN4: Design services for both gate-based quantum and annealing systems have been difficult to offer on the same platform and in an integrated manner.

LN6: Regarding the design of annealing-based systems, we believe that it is essential to offer a user-friendly interface beyond the definitions of the mathematical formulae, which can guide the user and control possible errors.

Despite all the components we have so far created and the great effort this has required, we still have a lot of work ahead of us if we are to turn the QuantumPath platform into a platform that truly supports software engineering. This work includes:

1. Integrating facilities for the re-engineering of quantum software,[34] which would allow on the one hand the transformation from classical to quantum software, and on the other the migration of quantum software from the gate-based paradigm to the annealing paradigm.
2. Developing a set of tools to assure quantum software quality, both as regards the testing and the metrics for quantum software quality (maintainability, performance, etc.).
3. Testing the platform with increasingly complex problems and assuring an agreed industry scale performance (as far as the underlying quantum hardware allows this).
4. Improving and tuning the telemetry system of QuantumPath, in order to evaluate quantum hardware and to optimise quantum programs that can be executed with high reliability on existing quantum hardware, following the proposal of Reference 35.
5. Incorporating in the platform some quantum error correction (QEC) tools[36] to deal with noisy bits and the problems they can cause.

## 7 | CONCLUSIONS AND FUTURE WORK

We are convinced that the development of the quantum industry will not have a future at a social scale until it moves beyond its current dependence on quantum scientists, mathematicians, and physicists, essential as they may be, since in its time the classical computing industry would not have had its current success had it depended only on cybernetic engineers.

As with classical computing, it will be software engineers and programmers, in interaction with users and the market, who will end up defining how and for what purposes quantum computing will be used. This will occur when they develop practical commercial applications that will shape first its usefulness and, progressively, its universality.

In recent years, quantum computing has experienced a breakthrough, with more and more companies taking up the challenge of designing and manufacturing quantum computers. The supply of tools for quantum software development is growing all the time, too.

Most of these resources are developed or promoted by quantum hardware vendors and are generally aimed at making it easier for users to interact with their own environments. But it is precisely this last detail that shows the most important limitation of working with proprietary environments. Since they are only valid for working in the ecosystems of the particular vendor, researchers and developers approaching a given quantum technology are obliged to learn how to work in the given environment in which they want to develop and execute the specific solution.

In this article, we have presented QuantumPath, a platform which is truly agnostic alternative, and which can enable multidisciplinary teams to focus on the design and development of quantum algorithms and processes, without the need to worry about a specific technology.

QuantumPath has been designed and developed in accordance with the principles and commitments of the Talavera Manifesto in the field of software engineering and quantum programming. Accordingly, in addition to being agnostic, it supports the management of quantum software development projects.

QuantumPath has been tested in recent months by a limited number of customers, invited from the health and finance sectors. Although it is not open-source software, it can be accessed in a very advantageous way if you are part of the aQuantum network,[‡] especially if you belong to a university or research centre.

In terms of future work, we are working on all the points mentioned in the previous section, as our goal is to create a complete quantum software engineering and programming platform.

## ACKNOWLEDGEMENTS

This work is part of 'QHealth: Quantum Pharmacogenomics Applied to Aging', 2020 CDTI Missions Programme (Centre for the Development of Industrial Technology of the Ministry of Science and Innovation of Spain) and FEDER. We would like to thank all the aQuantum members for their help and support. We also thank the anonymous reviewers whose comments and suggestions helped improve and clarify this article.

## AUTHOR CONTRIBUTIONS

All the three authors conceived and designed the tool and wrote the paper.

## DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data generated, or the article describes entirely theoretical research.

## ENDNOTES

*https://algassert.com/quirk

†https://www.youtube.com/watch?v=nrO4JzM5cWs

‡https://www.aquantum.es/partner-network/

## ORCID


*Mario Piattini* https://orcid.org/0000-0002-7212-8279

## REFERENCES

1. Lloyd S. *Programming the Universe: a Quantum Computer Scientist Takes on the Cosmos*. Vintage; 2006.
2. Deutsch D. *The Beginning of Infinity: Explanations that Transform the World*. Penguin Group; 2011.
3. Ding Y, Chong FT. *Quantum Computer Systems: Research for Noisy Intermediate-Scale Quantum Computers*. Morgan & Claypool Publishers; 2020.
4. Resch S, Karpuzcu UR. Quantum computing: an overview across the system stack; 2019. arXiv:1905.07240v1.
5. EQF. *Strategic Research Agenda*. European Quantum Flagship; February 2020.
6. Preskill J. Quantum computing in the NISQ era and beyond. *Quantum*. 2018;2:79. doi:10.22331/q-2018-08-06-79
7. Aaronson S. The limits of quantum computers. *Sci Am*. 2018;298(3):62-69.
8. Piattini M, Peterssen G, Pérez-Castillo R. Quantum computing: a new software engineering Golden age. *ACM SIGSOFT Softw Eng Newsl*. 2020;45(3):12-14.
9. Booch G. The history of software engineering. *IEEE Softw*. 2018;35(5):108-114.
10. Cusumano M. Technology strategy and management: the business of quantum computing. *Commun ACM*. 2018;61(10):20-22.
11. Zhao J. Quantum software engineering: landscapes and horizons; 2020. arXiv:2007.07047v1[cs.SE].
12. Piattini M, Peterssen G, Pérez-Castillo R, Hevia JL, Murina E. The Talavera manifesto for quantum software engineering and programming. QANSWER 2020 QuANtum SoftWare engineering & pRogramming. Proceedings of the 1st International Workshop on the QuANtum SoftWare Engineering & pRogramming, Talavera de la Reina, Spain; February 11-12, 2020. http://ceur-ws.org/Vol-2561/paper0.pdf
13. Piattini M, Serrano M, Pérez-Castillo R, Peterssen G, Hevia JL. Towards a quantum software engineering. *IT Prof*. 2021;23(1):62-66. doi:10.1109/MITP.2020.3019522
14. D-Wave Ocean Software Documentation. Accessed May 23,2021. https://docs.ocean.dwavesys.com/en/stable/
15. Fujitsu Quantum-Inspired Digital Annealer Services. Accessed May 23, 2021. https://www.fujitsu.com/global/services/business-services/digital-annealer/services/
16. IBM Quantum Experience Documentation. Accessed May 23, 2021. https://quantum-computing.ibm.com/docs/
17. Azure Quantum Documentation. Accessed May 23, 2021. https://docs.microsoft.com/en-us/quantum/
18. Forest SDK Documentation. Accessed May 23, 2021. https://pyquil-docs.rigetti.com/en/stable/
19. Strawberry Fields Documentation. Accessed May 23, 2021. https://strawberryfields.readthedocs.io/en/stable
20. PennyLane Platform Documentation. Accessed May 23, 2021. https://pennylane.ai/
21. Tools of Quantum Computing. Quantum Computing Report. Accessed May 23, 2021. https://quantumcomputingreport.com/tools/

22. Open-Source Quantum Software Projects. Accessed May 23, 2021. https://github.com/qosf/awesome-quantum-software
23. Orquestra. Product Overview. Accessed May 23, 2021. https://www.zapatacomputing.com/orquestra/
24. ProjectQ Web Site. Accessed May 23, 2021. https://projectq.ch/
25. Quantum Inspire. Accessed May 23, 2021 https://www.quantum-inspire.com/
26. QuantumPath. Accessed May 23, 2021. https://www.quantumpath.es/
27. Quantum Programing Studio. Accessed May 23, 2021. https://quantum-circuit.com/docs
28. Strangeworks QC Platform Documentation. Accessed May 23, 2021. https://strangeworks.com/platform
29. Hevia JL, Peterssen G, Ebert C, Piattini M. Quantum computing. *IEEE Softw*. 2021;38(5):7-15.
30. CTIC Quantum Computing. Accessed May 23, 2021. https://www.fundacionctic.org/en/computacion-cuantica
31. aQuantum, one of the seven star-ups chosen by CaixaBank to develop "Fintech" projects. Accessed October 2, 2021. https://www.aquantum.es/aquantum-one-of-the-seven-star-ups-chosen-by-caixabank-to-develop-fintech-projects/
32. QHealth. Quantum pharmacogenomics applied to ageing. Executive Summary. Accessed October 2, 2021 https://www.aquantum.es/rdi/qhealth/
33. Developing classical/quantum hybrid software with QPath. Accessed October 2, 2021. https://www.youtube.com/watch?v=Vo7atNN0AAg
34. Pérez-Castillo R, Serrano MA, Piattini M. Software modernization to embrace quantum technology. *Adv Eng Softw*. 2021;151:102933.
35. Murali P, Baker JM, Javadi-Abhari A, Chong FT, Martonosi M. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. ASPLOS'19: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems; April 2019:1015-1029.
36. Ball H, Biercuk MJ, Carvalho A, et al. Software tools for quantum control: improving quantum computer performance through noise and error suppression; 2020. arXiv:2001.04060.

**How to cite this article:** Hevia JL, Peterssen G, Piattini M. QuantumPath: A quantum software development platform. *Softw: Pract Exper*. 2022;52(6):1517-1530. doi: 10.1002/spe.3064