POLITECNICO DI MILANO

Computer Science and Engineering

Quantum Computing

State of the art Analysis

Software Engineering 2 Research Project

Academic year 2021 - 2022

08/04/2022

Authors:
Leonardo Gori,
Stefano Taborelli

Professor:
Elisabetta Di Nitto

1 Introduction

Quantum computing is an area still under strong development and research, which over the next few years may significantly impact the way we think about Software Engineering. Quantum Computing research is int the same stage as the development of classical computing was in the 60's. It is not possible to predict which will be the new technologies and discoveries that will revolutionize the way we are used to do research and they will impact the everyday life. The problems that are being encountered up to now are similar to those that have already been faced and, therefore, we can use the experience gained during the 900's to rethink the deployment and avoid some stage of development. Nowadays, technology development is mainly focused on quantum supremacy, where big tech companies are focusing their efforts to prove that they have created a hardware system that can outperform every other supercomputer currently built. With an article published in Nature, Google has declared to have built a 54-qubit processor that in 200 seconds was able to complete a benchmark for which the most advanced supercomputer would take 10000 years. But the technological development is not all based on pure computing power and benchmark, but also passes by other fundamental aspects. In the next sections we will explain the major criticalities present in the development of quantum software engineering and after we will dig into some of the new frontiers of development.

2 Quantum Software Engineering Problems

2.1 Programming Language

One of the most interesting directions to be addressed by quantum software engineering research is to propose adequate abstractions for modelling, designing, and building quantum programs, that allows programmers to overcome some of the main problem of developing quantum software. The quantum programming languages and paradigm that are used nowadays are not mature enough to allow programmers to develop software solutions for industrial applications and public services.

On the contrary, they are usually tightly linked to the hardware specifics of the machine the software is developed on, while the major amount of quantum software being developed is mainly made for research purposes and programs are low-level and made for solving specific instances of problems.

All these issues cause more effort for the companies that want to develop and implement quantum software for their business. All these problems point out analogous complexities already faced during the transition from sequential programming paradigm to the objected-oriented one, which maintained the same classic computing architecture, with a new way of think about the elements involved. Since the programming paradigm transition involves an additional change of the computing architecture, we expect that this step will introduce a higher level of complexity. In fact, non-Von Neumann architectures will require new techniques and tools to be programmed, since the principles on which quantum computers are based introduce new concepts as superposition, entanglement or decoherence. The idea of programming using sequential logic and objects is not anymore sufficient, the people involved need to think in a new quantum way.
Many companies along the world already started coining their own programming models and frameworks, specifically designed for their own machines, and optimized for few instances of problems. Among them, there are Qiskit, Cirq and Q# respectively from IBM, Google and Microsoft. This fact represents another obstacle when figuring out which computing infrastructure is best suited for developing a specific application, because each one of these frameworks has its own rules and structure. For these reasons, Quantum Software Engineering should also attempt to formalize a systematic approach for mapping the best programming infrastructure, based on a well-defined analysis of the characteristic the software solution should respect. In section 2.3, we present some of the main aspects that should be considered before moving on the development of the quantum software to be, in order to understand which solution can be best suited for its characteristics. Another solution to the same problem, may be also defining a unified standard quantum model, for the development of general-purpose programs, even if it seems a farther eventuality in the future.

2.2 Verification and Testing

Once developed, programs need to be tested and validated in order to obtain the expected results while guaranteeing a sufficient level of security and reliability. The verification and validation of the software represents a problem in quantum computing, since the measurement of a quantum bit state leads to the irreversible disruption of its superposition, preventing the possibility of its status changing in a subsequent moment. This represents a key point to be evaluated, since many of the most common testing practices in classical computing rely on the measurement of the status of the application while it is executing and changing its state. The most common techniques are debugging using breakpoints, printing status to evaluate the workflow and unit test, but all of them are unfeasible. Quantum software engineering should analyze new ways for testing and validating software.

2.3 Implementation and Integration

The quantum computers currently on the market are not made to replace the architectures we are using, but to be exploited to solve problems characterized by a strong parallelization and to solve NP problems that even today's supercomputers are not able to solve. The first important step is to identify the part of the software that could take the most advantage of the available computing power, but also the platform

on which run is a crucial aspect.
At the moment there is not any hardware dedicated to quantum computing available to large retailers, but the hardware implementation is in the hands of large technology companies that are still carrying out research in terms of improving performance, scalability, accuracy and the number of qubits available. Although the hardware is still undergoing by a strong growth and development, so the companies involved are providing some of their computing power that allows the development of many concrete applications, but not without some compromises. The development of the hardware has been divided into two distinct philosophies on how to develop the quantum mechanics that underlies the technology itself. Quantum gate aims to recreate the structure of current logic gate-based architectures and reuse all the experience gained over the past few decades in chip fabrication. While quantum annealing aims to optimize current problems by exploiting the characteristics of quantum mechanics. In addition to the different type of approach in hardware realization, each company proposes a different realization of the same concept to pursue its own development and research, making the choice between the various services even more complicated.

In order to be able to choose the Quantum Computing service that best suits your needs, the following points must be taken into consideration [Oli20]:

- Know how to consume the quantum services
- Know how the quantum mechanics characteristics are engineered in order to make it programmable in a reasonable way.
- Know which are the limitations of the computer, because we are at the beginning of the quantum era.
- Foresee the evolution of the provider in order to foresee potential changes in the investment we are doing trying to adapt the technology to our new solutions.
- Evaluate if it's available a simulator or we can access to real hardware.
- Know how much it cost

Once the provider is chosen, the QPSS (Quantum Provider Software Stack) is provided for the implementation of the customers, which it includes APIs, libraries, simulators and all the documentation needed to use it. The QPSS is proprietary, and this is a crucial aspect to take care during the implementation phase, because the chosen service can evolve quickly to comply with the limitations imposed by the provider for its own purpose or to follow the hardware evolution. This implies that the software a company wants to implement must be built to support continuous and unforeseen changes imposed by the service in order to keep running, otherwise huge amount of the budget will be dedicated to update it and to keep it running.

The final step is to integrate the quantum solution with the already present architecture. The hybrid architecture suggests having all benefit from the classical layers structure and a dedicated quantum node where to run the software that requires the most computational power. Nowadays this node has to be available through the internet and a dedicated networking module must be implemented, but, when the new hardware will be ready, the node could be integrated on site with the same approach.

3 Developer's Side

In the previous section, we considered which aspects influence the whole field of Quantum Software Engineering, whereas in this section we want to consider the critical issues that affect those who are approaching QSE for the first time.
Part of the research in this area has focused on analysing the problems and difficulties faced by developers when working on quantum projects. Shaydulin *et al* [STR20], conducted research on the background of programmers currently working on QSE projects, interviewing 148 developers on github and 46 experts from companies in the field. Their work revealed that one of the most critical issues is the number of people currently working on development in this field, compared to other SE fields, and most of them have an academic background other than computing, e.g. physics, mathematics or chemistry. Furthermore, they found only one person from a course dedicated to Quantum Computing, while most of the respondents are PhDs. This highlights the fact that there is a lack of a broader educational offer and a lower level of dissemination to allow more and more people to approach this field and thus bring about innovation.

In this paper, we attempt to define what could be the correct approach that a neophyte in the field of Quantum Software Engineering should follow in order to best approach this field of research. Our work extends what has been done by [STR20], which proposes the acquisition of theoretical issues, prior to software implementation. In this section, we report the path we took to best introduce ourselves to Quantum Software Engineering.

- Teaching

3.1 Teaching

As Booch reports 'No matter the medium or the technology or the domain, the fundamentals of sound software engineering will always apply: craft sound abstractions'. The cultural background required for the implementation of quantum software requires the mastery of a wide range of multidisciplinary skills, without which the subject cannot be understood.
Currently, the resources available for understanding the subject can be split between the content provided by outreach activities, such as YouTube videos or popular articles, and university-level teaching. The former are useful for providing an overview of Quantum computing to those who are not familiar with the subject. For our field of research, we decided to embark on the academic side by attending the only course offered by our university, a seminar offered by Cineca, and the precious work provided by Nannicini [Nan17]; all of them are related to the introduction to quantum computation.
The concepts that has been taught by this course are standard ones in the academic field: a basic introduction to Dirac notation, the concepts of entanglement and superposition and Quantum Gates. These allow an understanding of the basic concepts and how qbits work, but are not sufficient to guarantee complete autonomy for the implementation of quantum algorithms.

3.2

In contrast to classical computation, QSE requires an approach to algorithm's development that is mainly based on two fundamental points: a way of thinking and solving problems that is based on Quantum Circuits and the knowledge of probabilistic algorithms. In order to test the theoretical knowledge learnt and to understand this new paradigm, there are various free tools, grouped in [Com], which make it possible to experiment with Quantum Circuits quickly and easily.

3.3 Implementation

QSE is still at an embryonic stage of research, which is why a huge number of frameworks and programming languages are available, which are also featured in (Quantum Computing Report). Among the most widely used by the community are Qiskit, Q# and Cirq because they are widely supported and well documented. Furthermore, all of them offer tools for the implementation of algorithms based on the quantum gate model, and many features for the visualization of quantum circuits. The quantum gate model, in our opinion, has to be preferred w.r.t. the quantum annealing model when approaching the QC field for the first time, since it represents the *de facto* standard for a wider class of problems, and provides more support by the scientific community. For these reasons, users approaching the quantum software implementation for the first time, should choose one of the 3 aforementioned languages, according to their preferences and needs, to implement one of the algorithms presented in the literature.
In this paper, we propose the implementation of the Bennett-Brassard algorithm (BB84) for the distribution of public keys in a cryptographic system. We decided to use Qiskit for the following reasons:

1. It is based on python, which is more user-friendly because it is high-level and interpretative.
2. The simplicity of installation, which can be performed in several ways (i.e. pip install Qiskit).
3. It is supported by most available tools
4. The documentation is well written and the developer's community frequently supports the work and problems encountered by other people.

3.4 Simulation vs Real execution

There are two possibilities to execute the realised algorithm: simulation on a classical computer or execution on a quantum component in the cloud. Simulation on a classical computer is essential in order not to occupy quantum resources unnecessarily, without depending on the execution queue of the quantum machine or having to pay for execution. Having chosen Qiskit, we can simulate by choosing the hardware architecture that we will then use.

References

[Com] HKA Marketing Communications. Quantum Computing Report - Where Qubits Entangle with Commerce. https://quantumcomputingreport.com/tools/. Last accessed: May 2022.

[GFFA20] Cláudio Gomesa, Daniel Fortunatoc, João Paulo Fernandesa, and Rui Abreub. Off-the-shelf components for quantum programming and testing. In *International Workshop on Software Engineering & Technology 2020*, 10 2020.

[MBGAM20] Enrique Moguel, Javier Berrocal, Jose García-Alonso, and Juan Manuel Murillo. A roadmap for quantum software engineering: applying the lessons learned from the classics. In *International Workshop on Software Engineering & Technology 2020*, 10 2020.

[Nan17] Giacomo Nannicini. An introduction to quantum computing, without the physics, 2017.

[Oli20] Hevia Oliver. Requirements for quantum software platforms. In *International Workshop on Software Engineering & Technology 2020*, 10 2020.

[PCPPNHO20] Ricardo Pérez-Castillo, Mario Piattini, Guido Peterssen Nodarse, and Jose Luis Hevia Oliver. The quantum software engineering path. In *International Workshop on Software Engineering & Technology 2020*, 10 2020.

[STR20] Ruslan Shaydulin, Caleb Thomas, and Paige Rodeghero. Making quantum computing open. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. ACM, jun 2020.