An Empirical Study on the Current Adoption of Quantum Programming

Manuel De Stefano*
SeSa Lab - University of Salerno
Fisciano, Italy
madestefano@unisa.it

ABSTRACT

Quantum computing is no longer just a scientific curiosity; it is rapidly evolving into a commercially viable technology that has the potential to surpass the limitations of classical computation. As a result of this transition, a new discipline known as quantum software engineering has emerged, which is needed to describe unique methodologies for developing large-scale quantum applications. In the pursue of building this new body of knowledge, we undertake a mining study to elicit the purposes quantum programming is being used for, and steer further research.

ACM Reference Format:
Manuel De Stefano. 2022. An Empirical Study on the Current Adoption of Quantum Programming. In *44th International Conference on Software Engineering Companion (ICSE '22 Companion), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3510454.3522679

1 INTRODUCTION AND MOTIVATION

The dream has come true [18]: several physicists and computer scientists agree that quantum technology is right around the corner [17, 18] and that the 21st century will be recalled as the «quantum era.» [27]. Quantum computing promises to revolutionize program computation compared to classical computers [22], and offering an great polynomial speedup for certain problems[4, 23], and eventually achieving the so-called quantum supremacy [6]. For this reason, all major software companies, like IBM and Google, are currently investing hundreds of millions of dollars every year in quantum computing technologies[1], such as quantum programming languages, toolkits, and hardware.

While there have already been several promising applications of quantum programming in the fields of machine learning [8], optimization [15], cryptography [20], and chemistry [28], the development of large-scale quantum software seems to be still far from being a reality. In this respect, researchers [21, 25–27, 30] advocated the need for a new scientific discipline, the *quantum software engineering*(QSE) [30], which should allow programmers to develop quantum programs with the same confidence as classical programs by extending SE into the quantum domain.

Recognizing the effort of the research community, we noticed lack of empirical investigations to provide a complete view of the current *state of the practice* on QSE. In particular, we do not know how quantum programming is currently being used, which is critical to better understand the challenges faced by quantum developers and steer the future research.

To take a step in this direction, we propose an empirical study in which we mined all the GitHub repositories that employ the three most widely used quantum programming frameworks, i.e., Qiskit [5], Cirq [11], and Q♯ [2], and conducted a content analysis sessions [19] to elicit a taxonomy of tasks supported by quantum technologies nowadays.

During the first International Workshop on Quantum Software Engineering, researchers and practitioners have proposed a manifesto for quantum software engineering, known as the "Talavera Manifesto' [26].- which defines the set of fundamental principles of this new discipline. Some of these principles include agnosticism towards specific quantum technologies and coexistence of classical and quantum programming. Since then, several studies [7, 9, 12, 14, 21, 24, 25, 27, 31, 32] have been presented discussing challenges and potential direction in QSE research under various perspectives. The main potential research areas involve artifact modeling [7, 12, 14, 24], definition of software processes and methodologies for quantum programming [27], and quality issues [9, 31, 32].

2 APPROACH AND UNIQUENESS

The main *goal* of this preliminary study is to investigate the current usage of quantum programming technologies, with the *purpose* of understanding where the QSE research could bring benefits to the developers' community. The research question we aimed to answer was:

> **RQ.** *To what extent and for what kind of tasks are quantum programming frameworks being used?*

To answer it, we mined the open-source repositories hosted on GitHub which rely on quantum programming frameworks. We took into consideration only Qiskit, Cirq, and Q♯ since they are widely recognized as more mature than others [1, 3].

*Advisors: Andrea De Lucia, Fabio Palomba
[1]Boston Consulting Group report: shorturl.at/mINWY

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICSE '22 Companion, May 21–29, 2022, Pittsburgh, PA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9223-5/22/05. . . $15.00
https://doi.org/10.1145/3510454.3522679

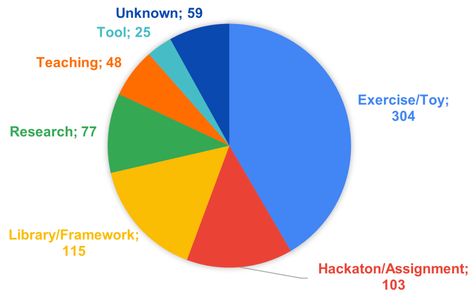

Figure 1: Partition of the repository per class of usage.

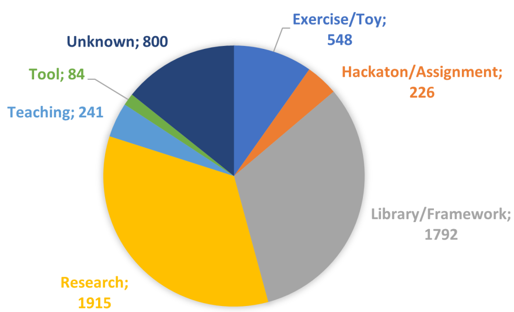

Figure 2: Number of contributors per type of repository.

Using the GitHub REST API[2], we looked for code snippets that indicated the use of the technologies we were interested in, namely **'from qiskit import'** and **'from cirq import'**, for QISKIT and CIRQ respectively. Q♯, on the other hand, is recognized by GITHUB as a programming language, and thus we looked for repositories using it as the primary language. Doing so, we found a total of 731 unique repositories (442 QISKIT, 217 CIRQ, 72 Q♯).

We then employed Straussian Grounded Theory [10], to correctly analyze the gathered data. The author (hereafter, *the main inspector*) manually analyzed each repository considering the README file and the repository description, looking for keywords that indicated the purpose of the repository, e.g., "My first Quantum Application", thus compiling the first classification. Then, two other inspectors (external collaborators working in the same research group) validated the initial labels and suggested how to improve them, e.g., by splitting or merging them.

In a second phase, the labels were renamed considering the feedback from the first step, thus grouping semantically similar [16] or even identical labels. This process was repeated until agreement over the labels,and achieving the theoretical saturation [29], i.e., the point when inspecting the labels offers no new insights.

Finally, the inspectors developed a taxonomy of the current usage of quantum programming based on the classification of the repositories. We also computed the agreement among the inspectors in terms of Fleiss' k [13], which resulted in a high agreement score of 0.993.

[2] PyGitHub: https://github.com/PyGithub/PyGithub

Table 1: Summary of the labels employed in the classification of the mined repositories

Label Name	Purpose
Exercise/Toy	Repository containing toy projects or collection of sample code.
Hackaton/Assignment	Repository containing code developed for a hackaton or a school assignment.
Library/Framework	Repository containing code composing a library or a framework.
Research	Repository containing code belonging to a paper or research appendix.
Teaching	Repository containing code that complements a lecture or a textbook.
Tool	Repository containing code for a tool.
Unknown	Repository not classifiable by reading the README or the Description

3 RESULTS AND CONTRIBUTION

Table 1 reports taxonomy of the current usage of quantum programming technologies, which is composed of six categories, representing the high-level purpose for which the repository was created. Figure 1 summarizes the repositories partitioned employing our taxonomy, whilst Figure 2 shows the distribution of developers per kind of repository.

Since quantum programming is still in its infancy, the main purpose of use is for exercise or personal study. The hosted code aims to explore the features of quantum programming and, in general, is not intended to become a real-world software. However, as shown in Figure 2 only 548 developers contribute to this kind of repositories, since most of these repositories have only one contributor. The other main purpose for which quantum programming is used is to develop quantum libraries or frameworks, which represent the 16% of the total repositories, and the second category for number of contributors. This was reasonably expected since quantum technologies currently under development are mostly open-source, and domain-specific libraries are also emerging (e.g., for quantum machine learning). Repository used as online appendices of research projects activities represent the 11% of the considered repositories, although having the greatest number of contributors. This result is in line with the fact that quantum programming is still a neat field in the vast plethora of computer science and physics research. The remaining cases (book appendices, blog posts, etc.) represent only a small percentage (7%) over the total.

In this paper we have only scratched the surface of quantum programming and QSE. Further research might involve a direct survey of the developers aimed to understand the challenges that thay face in conducting these tasks.

REFERENCES

[1] 2021. Open-Source Quantum Software Projects. https://quantumcomputingreport.com/tools/. Accessed: 2021-06-05.

[2] 2021. Q#: A Quantum Programming Lnaguage. https://qsharp.community. Accessed: 2021-09-21.

[3] 2021. What To Look For In A Quantum Machine Learning Framework. https://bit.ly/2ZkC0jr. Accessed: 2021-06-05.

[4] Scott Aaronson. 2005. Guest column: NP-complete problems and physical reality. *ACM Sigact News* 36, 1 (2005), 30–52.

[5] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, et al. 2019. Qiskit: An open-source framework for quantum computing. *Accessed on: Mar* 16 (2019).

[6] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510.

[7] Luis S Barbosa. 2020. Software engineering for'quantum advantage'. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 427–429.

[8] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (2017), 195–202.

[9] José Campos and André Souto. 2021. QBugs: A Collection of Reproducible Bugs in Quantum Algorithms and a Supporting Infrastructure to Enable Controlled Quantum Software Testing and Debugging Experiments. *arXiv preprint arXiv:2103.16968* (2021).

[10] Juliet M Corbin and Anselm Strauss. 1990. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative sociology* 13, 1 (1990), 3–21.

[11] Cirq Developers. 2021. *Cirq*. https://doi.org/10.5281/zenodo.4750446 See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors.

[12] Iaakov Exman and Alon Tsalik Shmilovich. 2021. Quantum Software Models: The Density Matrix for Classical and Quantum Software Systems Design. *arXiv preprint arXiv:2103.13755* (2021).

[13] Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76, 5 (1971), 378.

[14] Felix Gemeinhardt, Antonio Garmendia, and Manuel Wimmer. 2021. Towards Model-Driven Quantum Software Engineering. In *Second International Workshop on Quantum Software Engineering (Q-SE 2021) co-located with ICSE 2021*.

[15] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. 2017. Practical optimization for hybrid quantum-classical algorithms. *arXiv preprint arXiv:1701.01450* (2017).

[16] Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. 2015. Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies* 8, 1 (2015), 1–254.

[17] Tony Hoare and Robin Milner. 2005. Grand challenges for computing research. *Comput. J.* 48, 1 (2005), 49–52.

[18] Will Knight. 2018. Serious quantum computers are finally here. What are we going to do with them. *MIT Technology Review. Retrieved on October* 30 (2018), 2018.

[19] William Lidwell, Kritina Holden, and Jill Butler. 2010. *Universal principles of design, revised and updated: 125 ways to enhance usability, influence perception, increase appeal, make better design decisions, and teach through design*. Rockport Pub.

[20] Logan O Mailloux, Charlton D Lewis II, Casey Riggs, and Michael R Grimaila. 2016. Post-quantum cryptography: what advancements in quantum computing mean for it professionals. *IT Professional* 18, 5 (2016), 42–47.

[21] Enrique Moguel, Javier Berrocal, José García-Alonso, and Juan Manuel Murillo. 2020. A Roadmap for Quantum Software Engineering: Applying the Lessons Learned from the Classics.. In *Q-SET@ QCE*. 5–13.

[22] Leonie Mueck. 2017. Quantum software. *Nature* 549, 7671 (2017), 171–171.

[23] Masanori Ohya and Igor V Volovich. 2008. New quantum algorithm for studying NP-complete problems. In *Selected Papers Of M Ohya*. World Scientific, 83–90.

[24] Ricardo Pérez-Castillo, Luis Jiménez-Navajas, and Mario Piattini. 2021. Modelling Quantum Circuits with UML. *arXiv preprint arXiv:2103.16169* (2021).

[25] Mario Piattini, Guido Peterssen, and Ricardo Pérez-Castillo. 2020. Quantum Computing: A New Software Engineering Golden Age. *ACM SIGSOFT Software Engineering Notes* 45, 3 (2020), 12–14.

[26] Mario Piattini, Guido Peterssen, Ricardo Pérez-Castillo, Jose Luis Hevia, Manuel A Serrano, Guillermo Hernández, Ignacio García Rodríguez de Guzmán, Claudio Andrés Paradela, Macario Polo, Ezequiel Murina, et al. 2020. The Talavera Manifesto for Quantum Software Engineering and Programming.. In *QANSWER*. 1–5.

[27] Mario Piattini, Manuel Serrano, Ricardo Perez-Castillo, Guido Petersen, and Jose Luis Hevia. 2021. Toward a quantum software engineering. *IT Professional* 23, 1 (2021), 62–66.

[28] Markus Reiher, Nathan Wiebe, Krysta M Svore, Dave Wecker, and Matthias Troyer. 2017. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences* 114, 29 (2017), 7555–7560.

[29] Janiece L Walker. 2012. Research column. The Use of Saturation in Qualitative Research. *Canadian journal of cardiovascular nursing* 22, 2 (2012).

[30] Jianjun Zhao. 2020. Quantum software engineering: Landscapes and horizons. *arXiv preprint arXiv:2007.07047* (2020).

[31] Jianjun Zhao. 2021. Some Size and Structure Metrics for Quantum Software. *arXiv preprint arXiv:2103.08815* (2021).

[32] Pengzhan Zhao, Jianjun Zhao, and Lei Ma. 2021. Identifying Bug Patterns in Quantum Programs. *arXiv preprint arXiv:2103.09069* (2021).