

Quantum DevOps: Towards Reliable and Applicable NISQ Quantum Computing

Ilie-Daniel Gheorghe-Pop
Fraunhofer Institute for Open
Communication Systems (FOKUS)
Berlin, Germany
ilie-daniel.gheorghe-
pop@fokus.fraunhofer.de

Nikolay Tcholtchev
Fraunhofer Institute for Open
Communication Systems (FOKUS)
Berlin, Germany
nikolay.tcholtchev@fokus.fraunhofer.de

Tom Ritter
Fraunhofer Institute for Open
Communication Systems (FOKUS)
Berlin, Germany
tom.ritter@fokus.fraunhofer.de

Manfred Hauswirth
Technische Universität Berlin
Fraunhofer Institute for Open
Communication Systems (FOKUS)
Berlin, Germany
manfred.hauswirth@tu-berlin.de

Abstract— Quantum Computing is emerging as one of the great hopes for boosting current computational resources and enabling the application of ICT for optimizing processes and solving complex and challenging domain specific problems. However, the Quantum Computing technology has not matured to a level where it can provide a clear advantage over high performance computing yet. Towards achieving this "quantum advantage", a larger number of Qubits is required, leading inevitably to a more complex topology of the computing Qubits. This raises additional difficulties with decoherence times and implies higher Qubit error rates. Nevertheless, the current Noisy Intermediate-Scale Quantum (NISQ) computers can prove useful despite the intrinsic uncertainties on the quantum hardware layer. In order to utilize such error-prone computing resources, various concepts are required to address Qubit errors and to deliver successful computations. In this paper describe and motivate the need for the novel concept of Quantum DevOps, which entails regular checking of the reliability of NISQ Quantum Computing (QC) instances. By means of testing the computational reliability of basic quantum gates and computations (C-NOT, Hadamard, etc.) it consequently estimates the likelihood for a large scale critical computation (e.g. calculating hourly traffic flow models for a city) to provide results of sufficient quality. Following this approach to select the best matching (cloud) QC instance and having it integrated directly with the processes of development, testing and finally the operations of quantum based algorithms and systems enables the Quantum DevOps concept.

Keywords—Quantum DevOps, Quantum Computing, DevOps, Testing, Framework, IT

I. INTRODUCTION

The Quantum Computing (QC) domain entered the center stage in recent years due to the advancements of both hardware [1][2][3] and software platforms [4]. With its potential applications for boosting the optimal solution search for NP-hard problems in domains such as health, IoT, finance, security, energy, automotive, chemistry, AI, manufacturing, communications and infrastructure, QC may change information technology (IT) significantly.

Currently, QC evolves at great speeds from both hardware and software perspective. In the past 3 years, several new QC hardware platforms have become commercially available [5][6][7]. At the same time, open source software platforms and programming languages have emerged [4]. The goal is to bridge the gap between theoretical quantum mechanics (QM) and quantum hardware and software development so that QC becomes more feasible to use in real-world applications. Multiple obstacles exist in this process such as knowledge gaps between theoretical Quantum Mechanics and IT developers' skills and expertise, as well as the fact that the available hardware still has significant limitations. Moreover the software development, integration and testing processes are currently difficult.

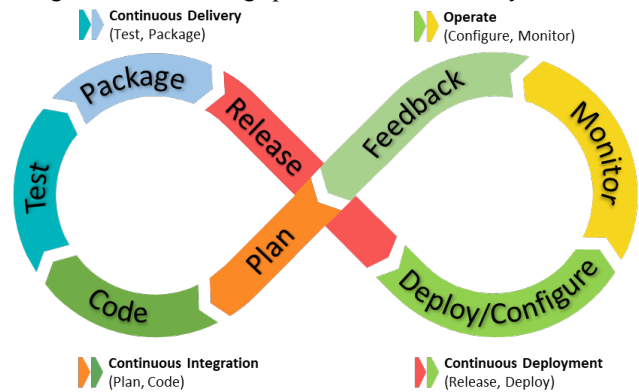


Figure 1: DevOps Flow Chart (adapted from [48])

Software development is expensive, especially in areas where only few standard platforms and tools exist. Thus, software development companies must strive to increase productivity and reduce time to market by reducing the time between software development and IT operations. This has driven the broad use of DevOps processes (Development & Operations) as shown in Figure 1. Based on the literature review and different understandings [8]-[13], we formulate our extracted perception of the DevOps concept as "An

emerging paradigm that emphasizes tight collaboration processes between Development and Operation departments within an IT company in order to achieve continuous delivery of reliable software". According to a 2017 survey [14], DevOps adoption has exceeded 50% within tech companies. Another report [15] suggests that the DevOps market growth is estimated at 20% up to 2026. In this paper we focus on QC software application development and operations, investigating the applicability and potential usefulness of a proposed Quantum DevOps concept.

This paper is organized as follows: Section II provides an overview on past and current work as well as developments in QC hardware, software and applications. Section III describes the current challenges identified in Quantum DevOps from a technical perspective. Section IV gives an overview of our approach to outline the necessary steps to establish a reference model for DevOps in QC. Section V provides details of the overall process architecture for developing, integrating, testing and operating Quantum Applications. Finally, in Section VI we draw our conclusions and provide an outlook on future work.

II. HISTORY AND RELATED WORK

QC was first mentioned around 1980 by Benioff [16], Menin [17] and Feynman [18] introducing quantum computation at a theoretical level. In 1988, the first physical realization of a quantum computer using Feynman's C-NOT gate was proposed [19].

In 1992, David Deutsch and Richard Jozsa proposed a quantum algorithm [20] for distinguishing between balanced and constant functions operating on a bit, thereby having an advantage over belonging classical algorithms. The next significant step was made in 1994, when Peter Shor developed his famous algorithm [21] which, in theory, can factorize large numbers in polynomial in $\log(N)$ time - i.e. $O((\log(N))^2(\log(\log(N)))(\log(\log(\log(N)))))$ which is a polynomial for the term $\log(N)$. This drew a lot of interest to the QC domain, as this algorithm could theoretically break encryption systems based on large prime numbers, e.g., RSA. Another quantum algorithm for database search proposed by Lov Grover [22] in 1996 offers a quadratic speed improvement over classical algorithms for unindexed databases. In 1998, Grover's search algorithm was demonstrated experimentally on a nuclear magnetic resonance machine (NMR) for the first time [23]. Between 2000 and 2010 quantum hardware based on different concepts were developed, from improving NMR machines to trapped ion technologies [24] and photonics [25].

Between 2011 and 2019, the first commercially available quantum computers entered the market following 2 main paradigms: quantum annealing [7][26] and universal models of quantum computing using superconducting electronic circuits [27] to implement discrete quantum gates [5][6]. Additionally, in 2019, Google announced having achieved quantum supremacy [28], by demonstrating a solution computed in 200 seconds on a quantum processor compared to solving the same problem in 10,000 years using a classical computer. Despite the various discussions around this result, it at least indicates the development direction of current QC approaches.

Current commercially available quantum computers include IBM [5], Rigetti [6], D-Wave [7], and IonQ [30]. In the research domain, experimental quantum resources are

utilized through national or international funding projects [35][36]. In terms of practical applications, several QC-based solutions have emerged. A rise in publications showcase quantum computing applied to molecular biology [37], nuclear physics [38], traffic optimization [39] and finance [40]. This trend confirms the potential of bringing this technology to market.

A big challenge to quantum computation which prevents immediately solving many of the big real-world problems consists of the noise and errors at the quantum gate level. John Preskill coined the name Noisy Intermediate-Scale Quantum (NISQ) for this technological era [29]. The current technology's sensitivity to interference, noise, and quantum de-coherence affecting quantum gate precision has sparked efforts to mitigate computational errors by both hardware [30][31][32][33] as well as software approaches [34].

From the software development perspective, apart from the infrastructure providers which have developed associated software platforms, such as IBM's Qiskit [41], Rigetti's Forrest [46] and D-Wave's Ocean [42], other companies have released software development kits (SDKs) such as Xanadu Penny Lane [43], Google Cirq [44], Microsoft Q# [45]. Additionally, there are also Open Source initiatives such as XACC [47] and many others (listed in [4]).

With regard to DevOps: In the classical computing domain, the development and delivery process of IT solutions is approaching maturity [6]. Although the context is different regarding operations and testing in quantum computing, we want to apply the DevOps foundational reference processes summarized by Leandro in [50] towards proposing Quantum DevOps. In the next section we outline the challenges of and the need for applying DevOps in the QC domain.

III. PROBLEM STATEMENT

We need to understand and accept that the Quantum Computing hardware will continue being unreliable and noisy in its calculations for the coming years – mainly due to the large complexity of the physical processes, the well-known problems such as quantum de-coherence times (when it comes to entanglement and relying computation), as well as due to various other aspects such as the required calibrations of the computing environment, uncertainties in the cooling temperature of the Qubits and potential errors in the controlling FPGAs and control software. In the coming NISQ era, quantum devices are expected to have enough stable Qubits (50-100) to be able to surpass traditional computers in some aspects for certain specific problems. However, the Qubits will not be able to perform reliably and all the computation will need to be intrinsically designed taking into account that the hardware layer is noisy and error-prone.

With this in mind, there is a clear need to provide the software tools and means for facilitating NISQ QCs in a way to enable the delivery of commercial Quantum Computations on top of an unreliable hardware layer. This challenge can be organized according to a number of research topics of paramount importance, with Quantum DevOps as one of the key tools to be researched.

IV. APPROACH

The main challenge, for which Quantum DevOps is required, arises from the uncertainties of NISQ QCs. On current QC architectures, even for relatively simple algorithms such as the Deutsch algorithm, we see that the computations can deliver completely different and even wrong results, when running the algorithm at different times on the same or on different architectures, e.g. depending on the number of Qubits, the cloud access, etc. Hence, there is a clear need to regularly check whether the prerequisites for a successful computation on a particular QC instance and the underlying architecture are fulfilled, in order to select the best QC instance and increase the chances for a critical optimization problem to be solved with the maximal possible correctness. The idea of Quantum DevOps can be summarized as:

- At regular intervals, various available QC instances are being checked for the calculation of basic gates.
- This provides an estimation of whether a QC instance is currently likely to be able to perform a large critical calculation correctly.
- Based on these checks, the most promising QC instance for a calculation is then selected (also among different cloud quantum providers).
- This process is applied in the development, testing and operations and merged into a kind of Quantum DevOps

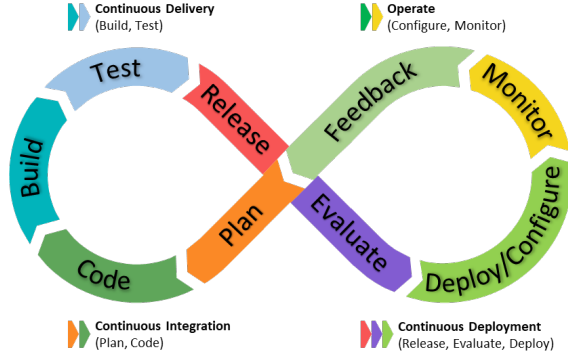


Figure 2: Quantum DevOps as an Extension of traditional DevOps

The Quantum DevOps is illustrated in Figure 2 as an extension of the traditional DevOps concept presented above. It consists of the planning (*PLAN*) and coding/programming (*CODE*) of a quantum algorithm (e.g. in Qiskit) and continues with the building/compiling/transpiling (*BUILD*) of the quantum code. This basic version of the developed algorithm is tested (*TEST*) in different environments, especially in simulations with or without added noise on the qubits. Given that the tests in the simulation environment (or in a controlled QC environment) were successful, the quantum software can be released (*RELEASE*), which means that the corresponding code is made available for deployment or execution on large scale cloud quantum computing platforms (such as IBM Quantum Experience). Thereby, the algorithm is triggered for a large critical computation (e.g. traffic optimization in a city) based on regular checks of the current performance of

different available quantum platforms and instances (*EVALUATE*), which after successfully passing the basic checks are candidates for running the large scale computation in question. Having selected the proper platform(s), the algorithm/system is deployed and configured (*DEPLOY/CONFIGURE*) and subsequently executed and monitored (*MONITOR*) such that resulting feedback from the execution process can be provided back (*FEEDBACK*) to the planning and development process in general. Hence, we see how the DevOps concept from traditional IT can be extended and adapted as to facilitate the integrated development and operations of quantum based systems in the coming NISQ era. This leads to known and established DevOps processes such as Continuous Integration (CI) and Continuous Delivery (CD) as illustrated in Figure 2. Based on these considerations, the following section details the main processes within the Quantum DevOps cycles.

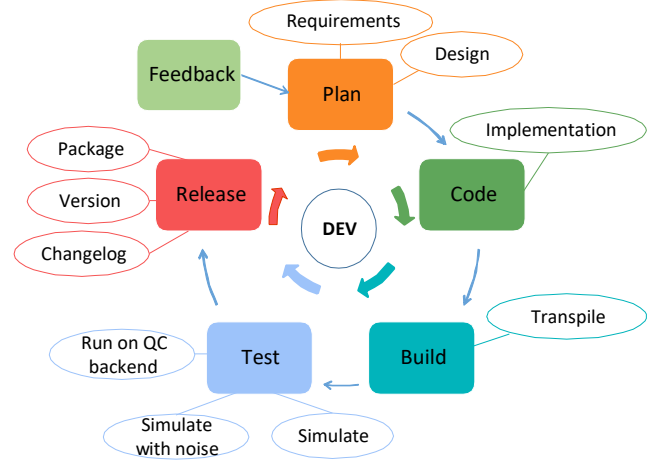


Figure 3: Details of the DEV Phase in Quantum DevOps

V. PROCESS DESCRIPTION

Figure 3 and Figure 4 present further details of the Quantum DevOps model and belonging reference processes. Figure 3 focuses on the Dev part which includes the following stages and belonging steps within the stages:

- **Plan** – within this stage the algorithm designer analyses the problem ahead and aims at capturing the technical and system requirements for the quantum based system/algorithm to be put in place. This results in a particular design for the approach towards the realization of the identified requirements.
 - *Requirements* – in this sub-step the requirements imposed on the system/algorithm have to be captured. The requirements can be managed with tools such as Doors, ProR and further from the domain of requirements engineering.
 - *Design* – during this sub-step proper design models must be derived, i.e. mainly system architecture and mathematical principles for the quantum algorithm/system to be designed. Typical

tools for this sub-step include the various *UML* and *SysML* tools on the market, the *Modelica* and *Matlab Simulink* based modelling software packages in addition to design tools such as the *Circuit Composer* of *IBM Quantum Experience*.

- **Code** – within this stage, the code for the belonging quantum based system/algorithm needs to be developed.
 - *Implementation* – the Implementation sub-step is responsible for the development of the code in a language of choice such as Qiskit or OpenQASM.
- **Build** – the current stage transforms the code in the corresponding language into specific instructions for the topology of the utilized simulation architecture or controlled QC test environment.
 - *Transpile* – the transpilation process is extremely important as to enable the mapping from the abstract circuit representation (e.g. in Qiskit) to the specific Qubit topology of the simulator or controlled QC test environment.
- **Test** – within this stage, the developed algorithm is subsequently tested in different environments.
 - *Simulate* – the test in a simulator in a perfect environment without any Qubit noise enables the developer to understand whether her/his approach functions in general and whether it is mathematically reasonable.
 - *Simulate with noise* – in this sub-step it is possible to understand whether the developed algorithm can function with some injected model noise that mimics a real Qubit environment. Hence, the robustness of the developed quantum system/algorithm to noise can be systematically examined and improved.
 - *Run on QC backend* – within this sub-step the system/algorithm is further tested on a real QC backend in order to improve it and evaluate it on a real quantum computer with all uncertainties of the NISQ era.
- **Release** – in case the above tests were successfully passed, the software has to be released, which includes the following steps:
 - *Package* – prepare an overall software package for the quantum based module/system.
 - *Versioning* – systematically provide a next version of the software release.
 - *Changelog* – compile and provide a changelog with belonging new features, bug fixes and ticket/defect IDs documenting the addressed aspects.

Thereby the above described stages interface at two places with corresponding stages from the Ops part as

illustrated in Figure 4. These are the **Release** stage of the Dev part interfacing and providing input to the **Evaluate** stage of the Ops part as well as the **Feedback** stage of the Ops part interfacing and providing input back to the **Plan** stage of the Dev part. These interactions clearly show where the Dev and Ops sub-processes are glued together, in order to provide the overall Quantum DevOps process. Hence, in order to complete the overall Quantum DevOps picture, Figure 4 depicts the stages and belonging steps of the Ops sub-process of Quantum DevOps. These include:

- **Evaluate** – this stage is at the heart of the Quantum DevOps and has been basically described in the previous section thereby formally consisting of the following steps:
 - *Evaluate QC Backend* – execute basic computation checks on relevant accessible cloud based quantum platforms (also including platforms from different providers). The checks should include computations based on the most important circuit gates (C-NOT, Hadamard, Pauli, Toffoli gates ...), in order to estimate the readiness and computational reliability of the candidate systems.
 - *Select QC Platform* – based on the evaluations in the previous step, the most suitable QC platform is selected to execute a required critical large scale computation based on the software developed in the Dev part of the Quantum DevOps process.
- **Deploy/Configure** – the software developed in the Dev part is prepared, configured and deployed on the most suitable QC platform according to the selection from the previous step.
 - *Transpile* – the software is once again transpiled for the characteristics and specifics of the most suitable target platform from the previous stage.
 - *Configure* – the software and the target hardware platform are configured and prepared for execution.
 - *Deploy* – the software is deployed and prepared for execution.
- **Monitor** – during this stage the software execution is monitored and information is gathered regarding the results and belonging execution properties.
 - *Monitoring Platform* – different hardware KPIs are monitored on the hardware platform during the execution (e.g. cooling temperature), stability of the Qubit entanglements ...
 - *Monitoring Execution* – the execution of the software is monitored in terms of number of shots, results distribution, number of iteration, execution times, and optimization properties of the algorithms etc.

- *Results Collection* – finally the results are collected and submitted as feedback to the Dev part.
- **Feedback** – within this stage all the gathered results from the monitoring are provided to the Plan stage of the Dev part in order to close the loop of Quantum DevOps. The communicated information can include but is not limited to aspects such as: Results Distribution, Optimization Rate, Execution Time ...

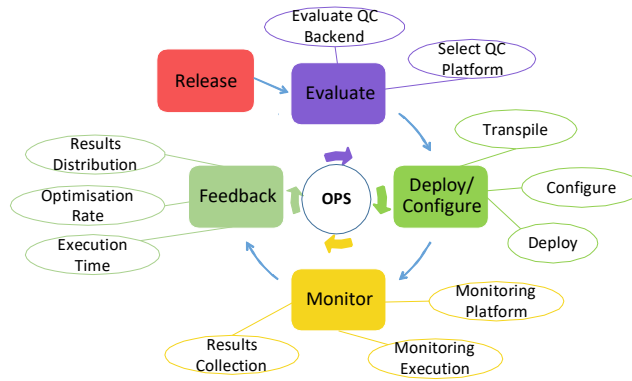


Figure 4: Details of the OPS Phase in Quantum DevOps

To summarize: based on the flows in Figure 3 and Figure 4 we can clearly observe how Quantum DevOps can be established as a process for the efficient development and production of critical and commercial quantum based systems in the NISQ era. We clearly see that this methodology has the potential to quickly deliver results within the development but also towards improving the execution of the belonging quantum modules in the face of the Qubit error rates of the NISQ era.

VI. CONCLUSIONS

The current paper motivated and presented the emerging concept of Quantum DevOps. Thereby, we presented our initial ideas and considerations which are going to be further refined and detailed in our research endeavor. In this line of thought, the motivation for Quantum DevOps lies in the current need to increase the number of Qubits in commercial and experimental QC systems whilst at the same time coping with the noise and errors in the Qubits and belonging gates.

Our experiences with currently available quantum computers show that even for relatively simple algorithms like the Deutsch–Jozsa, the results can vary strongly in time with respect to their correctness and precision depending on many aspects including de-coherence times, calibration requirements, temperatures of the cooling systems, potential issues with the control software and belonging FPGA modules. Hence, there is a clear need for a process that allows to regularly evaluate the available QC instances in terms of their computing capabilities and correctness before triggering a critical computation with potential significant impact in our everyday lives (e.g. the mobility and traffic related simulation towards optimizing routes or public transport schedules in a large Smart City). In order to achieve this, the processes of software/system development and operations need to be glued together with the belonging operational aspects allowing feedback and input to be

constantly exchanged between the various steps of the overall procedure. This intrinsically leads to the process of Quantum DevOps as described in this paper.

With regard to the core extensions that we conduct on the processes of traditional DevOps, we provide the following summary which is at the heart of the Quantum DevOps approach: 1) at regular intervals, the accessible QC instances are evaluated with respect to the calculation of basic gates and quantum computations, 2) the results of these regular evaluations are used to select the most promising (cloud) QC instance to execute the pending critical computation, 3) this process should closely relate and be applied to the overall process of development, testing and operations towards a Quantum DevOps.

Based on the above considerations, we present our updates to the steps and adopted/identified definition of traditional DevOps towards Quantum DevOps. Furthermore, we describe these steps in detail and draft the reference processes for enabling commercial applications even in the coming NISQ era with all its potential pitfalls. The detailed descriptions touch on topics such as requirements identification, system and algorithm design, transpiling, different types of simulation, packaging, versioning and further.

With respect to future work: we plan to expand and implement the concept and processes for algorithms of different complexity and involving different cloud quantum providers. Furthermore, we plan for thorough evaluations including different efficiency measurements and extensive numerical evaluations as to be able to provide an understanding regarding the efficiency and improvements provided by the proposed concepts.

REFERENCES

- [1] H. Huang, D. Wu, D. Fan, X. Zhu, “Superconducting Quantum Computing: A Review”, 2020, arXiv:2006.10433 [quant-ph]
- [2] Tristan Zaborniak, Rogerio de Sousa, “In Situ Noise Characterization of the D-Wave Quantum Annealer”, 2020, arXiv:2006.16421v1
- [3] [Online] Samuel K. Moore, “Honeywell Claims It Has Most Powerful Quantum Computer”, <https://spectrum.ieee.org/tech-talk/computing/hardware/honeywell-claims-it-has-most-powerful-quantum-computer>, retrieved 28.06.2020
- [4] [Online] Open-Source Quantum Software Projects, <https://github.com/qosf/awesome-quantum-software>, as of date retrieved 15.06.2020
- [5] [Online] Available: <https://www.ibm.com/quantum-computing/>, as of date 15.07.2020
- [6] [Online] Available: <https://rigetti.com/>, as of date 15.07.2020
- [7] [Online] Available: <https://www.dwavesys.com/services>, as of date 15.07.2020
- [8] S. K. Bang, S. Chung, Y. Choh, M. Dupuis, “A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps”, RIIT 2013, <https://doi.org/10.1145/2512209.2512229>
- [9] J. Wetzinger, U. Breitenbucher, F. Leymann, “Standards-based DevOps Automation and Integration Using TOSCA”, In IEEE/ACM International Conference on Utility and Cloud Computation, 2014
- [10] Smeds J., Nybom K., Porres I. “DevOps: A Definition and Perceived Adoption Impediments”, 2015. Lecture Notes in Business Information Processing, vol 212. Springer, Cham, LNBIP 212, pp. 166–177

- [11] M. A. McCarthy, L. M. Herger, S. M. Khan and B. M. Belgodere, "Composable DevOps: Automated Ontology Based DevOps Maturity Analysis," 2015 IEEE International Conference on Services Computing, pp. 600-607, doi: 10.1109/SCC.2015.87.
- [12] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. 2016. "What is DevOps? A Systematic Mapping Study on Definitions and Practices", Workshop Proceedings of XP2016, DOI:<https://doi.org/10.1145/2962695.2962707>
- [13] Onokoy, L., & Lavendels, J. (2018). Evolution and Development Prospects of Information System Design Methodologies, *Applied Computer Systems*, 23(1), 63-68. doi: <https://doi.org/10.2478/acss-2018-0008>
- [14] [Online] Robert Stroud, "2018: The Year Of Enterprise DevOps", <https://go.forrester.com/blogs/2018-the-year-of-enterprise-devops/>
- [15] [Online] Selbyville, Delaware, "DevOps Market growth predicted at 20% till 2026: Global Market Insights, Inc.", <https://www.globenewswire.com/news-release/2020/04/13/2015016/0/en/DevOps-Market-growth-predicted-at-20-till-2026-Global-Market-Insights-Inc.html> (accessed 24.06.2020)
- [16] Benioff, Paul (1980). "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines". *Journal of Statistical Physics*. 22 (5): 563–591. doi:10.1007/bf01011339.
- [17] Manin, Yu I (1980). "Computable and Noncomputable" Sov. Radio.
- [18] Richard P. Feynman, "Simulating Physics with Computers", 1982, *International Journal of Theoretical Physics*, Vol 21, Nos. 6/7
- [19] K. Igeta and Y. Yamamoto. "Quantum mechanical computers with single atom and photon fields." *International Quantum Electronics Conference* (1988)
- [20] David Deutsch and Richard Jozsa (1992). "Rapid solutions of problems by quantum computation". *Proceedings of the Royal Society of London A*. 439: 553–558. doi:10.1098/rspa.1992.0167.
- [21] Shor, P.W., "Algorithms for quantum computation: discrete logarithms and factoring". *Proceedings 35th Annual Symposium on Foundations of Computer Science*. (1994) IEEE Comput. Soc. Press: 124–134. doi:10.1109/sfcs.1994.365700.
- [22] Grover L.K., "A fast quantum mechanical algorithm for database search", *Proceedings, 28th Annual ACM Symposium on the Theory of Computing*, (May 1996) p. 212
- [23] Chuang, Isaac L.; Gershenfeld, Neil; Kubinec, Mark, "Experimental Implementation of Fast Quantum Searching", (April 13, 1998). *Physical Review Letters*. 80 (15): 3408–3411. doi:10.1103/PhysRevLett.80.3408.
- [24] Gulde, S; Riebe, M; Lancaster, G. P. T; Becher, C; Eschner, J; Häffner, H; Schmidt-Kaler, F; Chuang, I. L; Blatt, R (January 2, 2003). "Implementation of the Deutsch–Jozsa algorithm on an ion-trap quantum computer". *Nature*. 421 (6918): 48–50. doi:10.1038/nature01336.
- [25] O'Brien, J. L.; Pryde, G. J.; White, A. G.; Ralph, T. C.; Branning, D. (2003). "Demonstration of an all-optical quantum controlled-NOT gate". *Nature*. 426 (6964): 264–267. doi:10.1038/nature02054.
- [26] Kadowaki, T.; Nishimori, H. (1998). "Quantum annealing in the transverse Ising model". *Phys. Rev. E*. 58: 5355. doi:10.1103/PhysRevE.58.5355.
- [27] Clarke, John; Wilhelm, Frank K., "Superconducting quantum bits". *Nature*. 453 (7198): 1031–1042. (18 June 2008), doi:10.1038/nature07128
- [28] Arute, F., Arya, K., Babbush, R. et al. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 505–510 (2019). <https://doi.org/10.1038/s41586-019-1666-5>
- [29] Preskill, J., "Quantum computing in the NISQ era and beyond". *Quantum* 2, 79 (2018), <https://doi.org/10.22331/q-2018-08-06-79>
- [30] [Online] Available: <https://ionq.com/>, as of date 15.07.2020
- [31] A. Bermudez, X. Xu, R. Nigmatullin, J. O’Gorman, V. Negnevitsky, P. Schindler, T. Monz, U. Poschinger, C. Hempel, J Home, et al., "Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation," *Physical Review X*, vol. 7, no. 4, p. 041 061, (2017).
- [32] [Online] Samuel K. Moore, "Honeywell Claims It Has Most Powerful Quantum Computer", <https://spectrum.ieee.org/tech-talk/computing/hardware/honeywell-claims-it-has-most-powerful-quantum-computer>, visited 29.06.2020
- [33] Asaad, S., Mourik, V., Joecker, B. et al. "Coherent electrical control of a single high-spin nucleus in silicon", *Nature* 579, 205–209 (2020). <https://doi.org/10.1038/s41586-020-2057-7>
- [34] A. Holmesyz, M. R. Jokarz, G. Pasandix, Y. Dingz, M. Pedramx, F. T. Chong, "NISQ+: Boosting quantum computing power by approximating quantum error correction", (Apr 2020), arXiv:2004.04794v2
- [35] [Online] QuTech- advanced research center for Quantum Computing and Quantum Internet by TU Delft and TNO Available: <http://qutech.nl/>
- [36] [Online] Max Planck-UBC-UTokyo Center for Quantum Material Available: <https://www.fkf.mpg.de/en>
- [37] C. Outeiral, M. Strahm, J. Shi, G. M. Morris, S.C. Benjamin, C.M. Deane, "The prospects of quantum computing in computational molecular biology", *WIREs Computational Molecular Science*, (may 2020), arXiv:2005.12792 [quant-ph]
- [38] A. Roggero, A.C.Y. Li, J. Carlson, R. Gupta, G.N. Perdue, "Quantum computing for neutrino-nucleus scattering", *Phys. Rev. D* 101, 074038 (Apr 2020), doi:10.1103/PhysRevD.101.074038}
- [39] S.Yarkoni, F. Neukart, E.M. Gomez Tagle, N. Magiera, B. Mehta, K. Hire, S. Narkhede, M. Hofmann, "Quantum Shuttle: Traffic Navigation with Quantum Computing", (Jun 2020), arXiv:2006.14162 [quant-ph]
- [40] J. Cohen, A. Khan, C. Alexander, "Portfolio Optimization of 40 Stocks Using the DWave Quantum Annealer", (Jul 2020), arXiv:2007.01430 [q-fin.GN]
- [41] [Online] IBM Qiskit, Available: <https://qiskit.org/>, as of date 15.07.2020
- [42] [Online] D-Wave Ocean SDK, Available: <https://ocean.dwavesys.com/>, as of date 15.07.2020
- [43] [Online] Xanadu Penny Lane, Available: <https://www.xanadu.ai/software/>, as of date 15.07.2020
- [44] [Online] Google Cirq, Available: <https://cirq.readthedocs.io/en/stable/>, as of date 15.07.2020
- [45] [Online] Microsoft Q#, Available: <https://www.microsoft.com/en-us/quantum/development-kit>, as of date 15.07.2020
- [46] [Online] Rigetti Forrest, Available: <https://github.com/rigetti>
- [47] [Online] XACC Framework, Available: <https://github.com/eclipse/xacc>, as of date 15.07.2020
- [48] [Online] R. D. Vleeschauwer, The Garner DevOps Toolchain, <https://www.bluebridgesoftware.com/blog/26-devops-the-gartnertoolchain>, retrieved Jun 2020
- [49] M. I. Zarour. M. Alenezi, N. Alhammad, K. Alsarayrah "A Research on DevOps Maturity Models", (2019), DOI:10.35940/ijrte.C6888.098319
- [50] Sousa, Leandro; Trigo, Antonio; and Varajão, João, "DevOps – foundations and perspectives" (2019). CAPSI 2019 Proceedings. 8.