Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 238 (2024) 7–14

Procedia Computer Science

www.elsevier.com/locate/procedia

The 15th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 23-25, 2024, Hasselt, Belgium

Towards Continuous Development for Quantum Programming in Decentralized IoT environments

Michail Alexandros Kourtis[b], Nikolay Tcholtchev[a,*], Ilie-Daniel Gheorghe-Pop[a], Colin Kai-Uwe Becker[a], Georgios Xylouris[b], Evangelos Markakis[c], Matic Petric[a], Raphael Seidel[a] and Sebastian Bock[a]

[a]*Fraunhofer Institute for Open Communication Systems (FOKUS), Berlin, Germany*
[b]*National Center of Scientific Research "Demokritos", Athens, Greece*
[c]*Department of Electrical and Computer Engineering, Hellenic Mediterranean University, Heraklion, Greece*

Abstract: The progression in quantum computing and the rapid development of quantum computation hardware has raised expectations for its application to commercially relevant use cases in the future. However, the need for high-level quantum programming abstractions and targeted use cases paired with vertical applications, which can directly benefit from quantum computing, remains an open challenge. This paper presents our vision for a decentralized architecture for swarm based IoT systems that leverages a high-level continuous development and integration quantum programming suite to support edge processing capabilities for different use cases across the edge-fog-cloud continuum. The planned Quantum DevKit provides the necessary abstractions and low-level backend interfaces for quantum computing infrastructure, enabling edge computation using quantum processing, with extensions to efficient management of Service Level Agreements (SLAs). The paper focuses on the presentation of the development kit and its coupling swarm-based architecture that automates the orchestration of the cloud-to-edge continuum, showcasing the potential of quantum technology in edge processing.

Keywords: Quantum Programming; Swarm Intelligence; Decentralized Computing; Edge; Blockchain

* Corresponding author. Tel.: +49-30-3463-7175.
E-mail address: nikolay.tcholtchev@fokus.fraunhofer.de

10.1016/j.procs.2024.05.190

1. Introduction

The integration of quantum computing with decentralized IoT swarm intelligence has the potential to transform various industries by enabling novel and sophisticated approaches to solving complex problems. With the advent of quantum computing, the way we perceive and solve problems is undergoing a major shift [1]. The unique properties of quantum computing, such as superposition and entanglement [2], enable it to tackle problems that are beyond the reach of classical computers [3]. Decentralized IoT swarm intelligence, on the other hand, has the potential to provide a novel approach to problem-solving by harnessing the collective intelligence of a large number of connected devices [4] across a wide-area IP network such as the Internet.

The following subsections outline the various aspects (e.g., data access) to be considered towards designing a highly distributed edge-fog-cloud computing environment with the possibility to integrate quantum computing resources.

1.1. IoT Swarm Intelligence and Quantum Computing

Leveraging the computational prowess of quantum computing with the adaptability of decentralized IoT swarm intelligence has the potential to establish an intelligent, highly efficient system capable of addressing diverse challenges, e.g., Artificial Intelligence [5], Cybersecurity [6], across various use cases and applications. This paper delves into the merits of integrating these two powerful technologies. The fusion of quantum computing and IoT swarm intelligence allows for the creation of IoT systems that are not only self-organizing, but also capable of making more efficient, real-time decisions. The quantum-based approach enables these systems to seamlessly manage and navigate through dense network environments, optimizing performance even in the face of potential degradation or failure. The enhanced decision-making capabilities also help accelerate the speed of service recovery, ensuring the uninterrupted execution of Service Level Agreements (SLAs). Moreover, the scalability of quantum computing can accommodate the increasing complexity and volume of data in IoT systems, providing comprehensive solutions tailored to specific use cases. In essence, this paper examines the transformative potential of combining a quantum computing development kit and an IoT swarm-based architecture, emphasizing the practical applications and the subsequent benefits that such a system could bring to multiple sectors.

1.2. Access to Data

Another gap to be addressed is related to the ease of access from the side of the data scientists and engineers across the edge-fog-cloud continuum. Public clouds already provide user-friendly abstractions (notebooks, simplified administration interfaces, graphical workflow designers, etc.) to data experts, so that the latter can concentrate on the management of the data and the selection and optimization of the ML/AI algorithms, rather than on the management of the physical and virtual resources which are needed and committed. This is a feature currently missing within edge orchestration solutions.

1.3. Quantum Programmability in Edge-Fog-Cloud

It is crucial that quantum computers can be controlled and programmed using high-level languages like traditional programming, rather than focusing on hardware details. The issue of lacking high-level abstractions for programming quantum computers has been acknowledged and several high-level languages have been proposed. However, many still require manual input of qubits and gates [7], [8], [9] or don't have industrial relevance since they are purely theoretical with no means of connecting to actual quantum hardware [10], [11].

The goal of the proposed work is to integrate Qrisp [12], [13] and our Quantum DevOps concept [14], in order to address these gaps and provide a framework with: 1) a uniform programming interface and abstraction, 2) simple and efficient syntax, and 3) accessibility for executing quantum algorithms on physical backends.

Qrisp is high-level programming framework for Quantum Computing that is written in Python and its API source code is also in Python, providing access to the vast Python library ecosystem. The output of Qrisp programs is circuit objects, making them compatible with various physical quantum backends and simulators. Quantum DevOps

[14] stands for an integrated process for the development and operations of quantum-based systems. This process extends the traditional DevOps ideas as established within the IT industry. Correspondingly, we plan to utilize Quantum DevOps and its belonging tools and tool chains for the overall purpose of efficiently programming quantum and hybrid algorithms over the edge-fog-cloud continuum.

1.4. The OASEES Vision

OASEES stands for *"Open autonomous programmable cloud apps & smart sensors"* and is a HORIZON Europe project that has recently started [15]. The OASEES project plans to deliver a European, fully open-source, decentralized, and secure Swarm programmability framework for edge devices and leveraging various AI/ML accelerators (FPGAs, SNNs, Quantum) while supporting a privacy-preserving federation process. Hence, the OASEES vision is to provide the open tools and secure environments for swarm programming and orchestration for numerous fields, in a completely decentralized manner. An important aspect in this process is identification and identity management, in which OASEES targets the implementation of a portable and privacy preserving federation system, for edge devices and services, with full compliance and compatibility to GAIA-X federation and IDSA (International Data Spaces Association) trust directives and specifications. This situation solidifies the need for an integrated enabler framework tailored to the edge's extreme data processing demands, using different edge accelerators, i.e., GPU, NPU, SNN and Quantum Computing.

Over the past years, several platforms, including open-source ones, have emerged, focusing mainly on the management and orchestration of edge infrastructure and services. However, to fully exploit the potential of edge processing, there is a need for a more holistic solution, embracing the entire edge-fog-cloud computing continuum, including central infrastructures (public clouds and networks) as well as smart devices. While it is recognized that the future of cloud computing will be distributed and heterogeneous, there is a lack of open management frameworks to address this dispersion and heterogeneity. Commercial solutions for hybrid core/edge management, such as Azure Edge Stack, indeed exist, but these are closed, restricted to specific deployment scenarios and only provided as managed services, i.e., not suitable for private (fully on-premises) deployments. Hence, the OASEES vision is to address these issues by providing open frameworks for programming and orchestrating different computing services and back-ends in a highly distributed edge-fog-cloud environment.

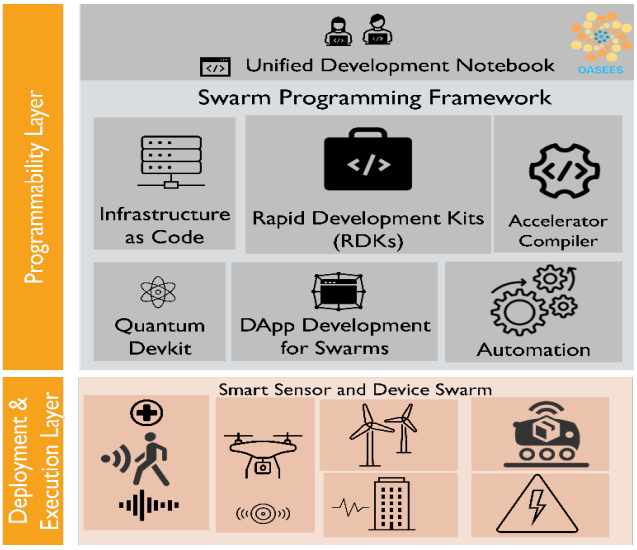

Figure 1: OASEES programmability layer architectural components.

2. OASEES Architecture

2.1. OASEES System Concept

The OASEES stack is expected to enable the user to i) discover and select available platforms, services and capabilities pertinent to their needs; ii) develop AI and Quantum services as well as automate lifecycle management operations; iii) deploy and manage AI and Quantum workflows across the edge-fog-cloud compute continuum; iv) configure service performance constraints; v) interactively explore data and exercise MLOps and Quantum DevOps; vi) verify the integrity of infrastructure and services across the continuum.

2.2. Programmability Layer

The Programmability Layer of the envisioned OASEES architecture is depicted as part of Figure 1, which also contains the corresponding Deployment & Execution Layer relating to the various devices, on which the distributed swarm intelligence programs can be executed. In the following, we present the key components of this layer.

Unified Development Notebook OASEES

OASEES plans to provide a graphical User Interface/Dashboard, mainly targeting at lifecycle management operations. Through this OASEES UI (User Interface), end users are able to express their requirements, select and deploy service templates (matching their requirements), configure data sources and sinks, monitor and terminate their services.

The OASEES SDK is built on top of an extensible set of development modules abstracting different parts of the IoT-to-edge-to-cloud spectrum, and results into a set of modules building upon them. Interaction between components follows a message-passing paradigm (e.g., pub/sub architecture). Each of the services and components implements the interaction with particular hardware or functionality, i.e., interacting with the APIs of a different cloud, network and/or edge systems. The OASEES orchestration framework is responsible for abstracting the interaction with cloud APIs (e.g., OpenStack, Kubernetes, KubeEdge …), the network provision module will abstract network system APIs (e.g., cloud-fog-edge interaction), and blockchain (EVM=Ethereum Virtual Machine, IoTA). Enablers for data-rich services in OASEES are dedicated hardware accelerated endpoints for the management and interaction with AI and quantum acceleration hardware as well as with smart devices (via dedicated agents). The AI acceleration and smart device interfacing abstracts the APIs as provided by highly specialized acceleration hardware such as FPGAs, AI ASICs, Quantum Processing Units, etc.

RDKs (Rapid Development Kits)

In addition to the Notebook approach, and complementing it, the OASEES SDK provides a suite of tools that enables developers to create edge applications tailored to their needs. Developers are able to select from pre-configured connectors and templates for faster integration and connectivity across their existing infrastructure. The SDK will provide libraries in at least two languages (e.g., Python and Go) for interfacing with the OASEES computing infrastructure, as well as data management libraries to accelerate the creation of distributed ML apps. An App Developer Guide will be provided, containing guidelines, patterns or tutorials. The abstraction of the underlying hardware and capabilities accelerates development and facilitates interoperability and portability.

Quantum Devkit

Quantum acceleration and processing for distributed computing and IoT services/applications still remain open challenges and opportunities for different fields. Currently, a noteworthy issue in the pathway towards wider adoption remains the communication and data exchange, between quantum systems and vendor cloud and edge frameworks.

In this regard, OASEES aims to develop Quantum APIs along the edge-fog-cloud continuum, targeting the following stages: i) Realization of a high-level programming language for Quantum Computing - programming without gates and qubits, ii) Integration of the higher-level programming language with the Quantum APIs along the edge-fog-cloud continuum, iii) Realization of hybrid Quantum Machine Learning algorithms on the base of the high-level programming language for Quantum Computing, iv) Quantum DevOps tools for the systematic and Model-Driven Development and Operation of quantum computing algorithms. In this respect, Fraunhofer FOKUS plans to integrate its existing access to Quantum Processing Units (QPUs) and will delve into the integration of a Quantum Devkit for swarm applications and services at the edge.

DApp Development for Swarms

A DApp (Decentralised Application) implementation framework consists of three key building elements, which we dubbed the DApp triplet in OASEES. These elements are (i) a trusted decentralized ledger - i.e., network connecting a swarm in the OASEES case, (ii) trusted decentralized execution of program logic, and (iii) decentralized applications. An example of a DApp triplet is a DAO (Decentralized Autonomous Organization) network, smart contracts (SC) implemented in this network, and blockchain enabled front end applications, which provide user interfaces and run embedded IoT devices and edge/cloud accelerators. The DApps to be developed in OASEES support natively the smart contract inference for different use case scenarios and parameters. Different technology

monitoring endpoints consume heterogeneous raw data (health vital sensors, building monitoring, electric vehicle observation, etc.) and different events and alarms are triggered automatically through the smart contract adoption. DAO in this case also involves the Human-in-the-Loop decision making where for sensitive and critical events a specialist is to be involved, i.e., medical practitioner, civil engineer, etc. Overall, this module involves the convergence of different technologies and tools from blockchain, e.g., Remix, to edge device programming, e.g., Node-RED.

Edge Package Rollup

This module focusses on the packaging of the DApp to be deployed across the entire swarm in an efficient manner with its respective security. Its aim is to provide a step-by-step workflow to minimize the risks of deploying decentralized behaviours: a low-resource simulator to iterate quickly on the design and test thousands of units, followed by a more realistic full-stack software-in-the-loop environment, extended whenever available to hardware-in-the-loop validation and finally to the deployment of the behaviour in the field. Additionally, regarding the security and fortification of this process, the Zero Knowledge (ZK)-Rollups - a paradigm from the blockchain sector – is planned for integration. ZK-Rollups vastly reduce the computing and storage resources required to validate blocks, by decreasing the amount of data in a transaction. This is made possible through cryptographic zero-knowledge proofs (ZKPs), whereby a party can prove that they know or have something without providing any information about what it is they know or have.

3. Quantum Development Kit

The main parts of the Quantum DevKit consist of the overall Quantum DevOps process, the Qrisp programming language, a Quantum Benchmarking approach, which will be integrated in the overall development and operations process, in addition to a set of different tools that will be experimented with and deployed during OASEES.

3.1. Quantum DevOps

The Quantum DevOps process cycle was initially defined in [14] and constitutes an extension of the DevOps concept as known from traditional software development. The structure of Quantum DevOps with its belonging phases is depicted in Figure 2. A detailed description with all involved steps/phases from the perspective of Quantum Computing is also provided in [14].

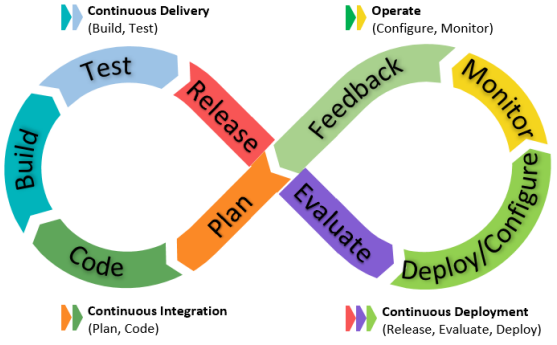

Figure 2: The Quantum DevOps concept as presented in [14].

The typical phases of PLAN (e.g., gather requirements, design and specification), CODE (implementation), BUILD and TEST are accompanied by quantum specific activities such as the mathematical design and simulation of a quantum algorithm (e.g., in a state-vector simulator) extended by the simulation with noise, given that all current quantum computers are NISQ (Noisy intermediate-scale quantum) devices, which perform calculations with a solid degree of noise and uncertainty. Furthermore, the TEST phase would also involve the testing on real noisy devices as to understand how robust the developed algorithm would be in the real world. The operations part of the cycle is extended by an EVALUATE phase in Figure 2 as compared to traditional DevOps principles. The EVALUATE step can be seen as a pre-deployment phase, in which available QPUs are checked with respect to their expected performance (e.g., fidelity, noise ratios, decoherence times) for the deployment of the quantum/hybrid-based system/algorithm in question. Based on this pre-evaluation, the QPU is selected, which is most likely to deliver satisfactory results for the planned execution. Afterwards the DEPLOY, MONITOR and FEEDBACK phases lead through the execution of the quantum algorithm and to the acquisition of results and performance indicators, which are subsequently

contributed to the PLAN step of the development sub-cycle, thereby closing the overall Quantum DevOps loop.

Through the introduction of such integrated development, operations and monitoring processes, we hope to improve the performance of quantum and hybrid systems in the near future and to bring them closer to real world use cases and deployments.

3.2. Qrisp High-Level Programming Framework

The Qrisp framework [12], [13] for high-level programming of quantum computers is at the heart of the Quantum DevKit for the OASEES swarm intelligence platform. Qrisp aims at simplifying and abstracting the current assembler-like practices for quantum programming.

```
from qrisp import QuantumFloat

qf_0 = QuantumFloat(3, -2)
qf_0[:] = 0.5

qf_1 = QuantumFloat(3, -1)
qf_1[:] = 1.5

result = qf_0*qf_1
print(result)
{0.75: 1.0}
```

Code 1: Multiplication of two Non-integer numbers with Qrisp. The `print(result)` command executes the program on a quantum simulator. Thus, the output is 0.75

As one can see from the code below, Qrisp provides abstract data types and hides the complexity of dealing with qubits, gates as well as resource (e.g., Uncomputation, automatic oracle generation [17]) and instruction management on the lowest level as currently done in established quantum programming frameworks/languages [18], [19].

Code 1 below shows the multiplication of two floats and the automated measurement of the result and its printout, without having to explicitly deal with the qubits and their topology as well as with required gates and rotations to implement a floating-point number multiplication procedure [20].

The abstract representation of Qrisp code takes away the complexity from the quantum programmer and moves it to the compiler/transpiler, which has to be implemented for different QPUs with their topology, architecture and specific physical principles and characteristics. However, this is also the case with high-level programming languages in the context of traditional computing and strongly increases the aspect of technological sovereignty since quantum algorithms are programmed once in an abstract manner and then specifically translated to the right QPU. This enables the diversification of the algorithms' implementation with respect to the backends of different vendors and makes the migration between quantum backend providers/vendors easier, thereby remediating the potential issue of a vendor lock-in.

3.3. Quantum Benchmarking

Quantum Benchmarking is another vital aspect that should be provided as a service in the context of the OASEES eco-system for highly distributed computing and swarm intelligence. Thereby, OASEES plans to provide edge services for evaluating QPUs and conveying this information to potential quantum/hybrid systems and algorithms. In addition, it should be possible to judge on the suitability of a particular QPU and the belonging software stack and hardware components for a specific algorithm. This requires considering various aspects – depicted in Figure 3 - that can influence the performance of an algorithm on a particular QPU and belonging software stack. The different factors such as SPAM (State Preparation and Measurement) fidelity, qubit topology, previous jobs in the queues and others are presented in a layered structure in [20]. These different factors are considered in current standardization efforts [21] towards capturing fair, abstract and transferable metrics for conducting quantum benchmarking.

3.4. Quantum Development Tools

Different tools are planned for integration into the Quantum DevKit, in order to realize the Quantum DevOps cycle. For example, we plan to investigate various coding editors such as *Eclipse*, *PyCharm* and *Spyder* and their possible integration in an overall web-based development kit.

An important phase in DevOps are the "building" and "coding" steps, for which various tools can be used. An important aspect in this context is the need for source code management and versioning, usually done with *git* or a git-based service. In addition, package management - which basically enables the preparation of "runnable" solution artefacts - plays a crucial role in the processes of building quantum computing solutions and systems from developed code. In this respect, it is also highly relevant to the "release" step of Quantum DevOps. Common tools for managing code packages are: *JFrog Artifactory*, *Npm*, *Docker Hub*, *Yarn*, *NuGet* and *Azure Artifacts*. Furthermore, the automation of *build*, *compilation* and *integration* that precedes the release process can be performed by tools such as *Gradle*, *ANT* and *Maven*.

The "testing" step is another extremely important phase in the development sub-cycle of Quantum DevOps in Figure 2. Various tests can be run during development so that potential bugs can be detected and fixed early. In the course of the development process, it is possible to implement various module tests, e.g., with *PyTest*. Other frameworks for defining and executing tests in this phase include *Junit*, *Cucumber*, *Jasmine*, *Rspec* and *Nunit* and can be used depending on the interface (e.g., REST or native) between the programming environment and the quantum backend or simulator. The associated test configurations also need to be managed, which is often done by frameworks such as *Serverspec* and *Chef-InSpec*. Finally, the performance of a quantum computing solution/system can also be measured initially during the development process to identify potential future bottlenecks, using various frameworks such as *Jmeter*, *Blazemeter*, *Locust* and *K6*. The above discussions only draft a couple of tools which could be of consideration for integrating them to a toolchain realizing the Quantum DevKit in the OASEES platform. However, this list is by far not exhaustive and does not even cover all Quantum DevOps steps. Hence, many more tools and integration efforts are planned during the development and experimentation phases of the OASEES project.

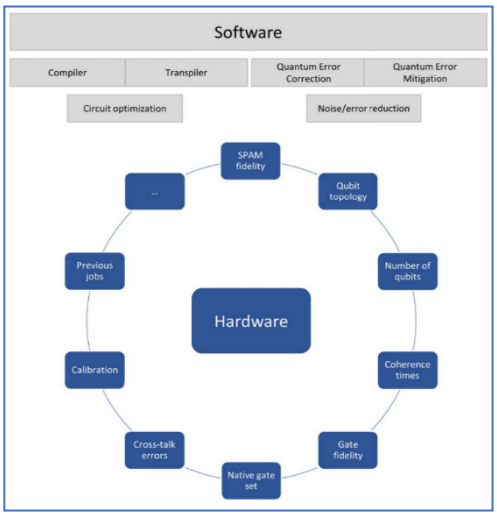

Figure 3: Different factors influencing the Quantum

4. Conclusions

The current paper presented our research and development activities towards the realization of a highly distributed computing platform for swarm intelligence over the edge-fog-cloud continuum. This platform is under development in the course of the recently started HORIZON Europe OASEES project and integrates among others a Quantum DevKit, which is meant to facilitate the development of quantum-hybrid algorithms, in order to accelerate and support algorithms for swarm intelligence. Hence, the presentation of the paper focused on the quantum programming aspects. This includes the Qrisp high-level programming framework/language to be adapted for the purposes of distributed development and execution of quantum algorithms integrating with other accelerator platforms such as GPUs, ASICs, FPGAs, neuromorphic chips and others. Furthermore, the goal is to integrate the Python based framework Qrisp in an overall Quantum DevOps process. Quantum DevOps is a concept extending the traditional DevOps by means and components, which are especially targeting the execution of quantum algorithms in NISQ noisy environments.

Acknowledgements: The research leading to these results has been supported by the OASEES project (no. 101092702).

References

[1] M. Bhatia and S. K. Sood, "Quantum Computing-Inspired Network Optimization for IoT Applications," in IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5590-5598, June 2020, doi: 10.1109/JIOT.2020.2979887.

[2] J. Preskill, "Quantum Computing in the NISQ era and beyond," Quantum, 2, 79 (2018).

[3] R. Raz and A. Tal. 2022, “Oracle Separation of BQP and PH,” in J. ACM 69, 4, Article 30, August 2022, 21 pages, doi: 10.1145/3530258

[4] Rath, Mamata, Darwish, et al.,”Swarm intelligence as a solution for technological problems associated with Internet of Things”, 2020, Swarm Intelligence for Resource Management in Internet of Things, Elsevier, p. 21-45

[5] N. Abdelgaber and C. Nikolopoulos, "Overview on Quantum Computing and its Applications in Artificial Intelligence," 2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Laguna Hills, CA, USA, 2020, pp. 198-199, doi: 10.1109/AIKE48582.2020.00038.

[6] El-Latif, Ahmed A. Abd, et al., “Quantum-Inspired Blockchain-Based Cybersecurity: Securing Smart Edge Utilities in IoT-Based Smart Cities”, 2021-07, Information Processing &ampmathsemicolon Management , Vol. 58, No. 4, Elsevier BV, p. 102549.

[7] K. Svore, M. Roetteler, A. Geller, M. Troyer, J. Azariah, C. Granade, B. Heim, V. Kliuchnikov, M. Mykhailova, and A. Paz, “Q#: Enabling Scalable Quantum Computing and Development with a High-level DSL,” Proceedings of the Real World Domain Specific Languages Workshop 2018 on - RWDSL2018. ACM Press, 2018. [Online]. Available: https://doi.org/10.1145%2F3183895.3183901

[8] A. Abhari, A. Faruque, M. J. Dousti, L. Svec, O. Catu, A. Chakrabati, C.-F. Chiang, S. Vanderwilt, J. Black, F. Chong, M. Martonosi, M. Suchara, K. Brown, M. Pedram, and T. Brun, “Scaffold: Quantum programming language,” 07 2012.

[9] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, “Quipper: A scalable quantum programming language,” SIGPLAN Not., vol. 48, no. 6, p. 333–342, jun 2013. [Online]. Available: https://doi.org/10.1145/2499370.2462177

[10] B. Bichsel, M. Baader, T. Gehr, and M. Vechev, “Silq: A high-level quantum language with safe uncomputation and intuitive semantics,” in Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, ser. PLDI 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 286–300. [Online]. Available: https://doi.org/10.1145/3385412.3386007

[11] F. Voichick, R. Rand, and M. Hicks, “Qunity: A unified language for quantum and classical computing,” 2022. [Online]. Available: https://arxiv.org/abs/2204.12384.

[12] R. Raphael Seidel, Sebastian Bock, Nikolay Tcholtchev, and Manfred Hauswirth. “Qrisp: A framework for compilable high-level programming of gate-based quantum computers” In PlanQC - Programming Languages for Quantum Computing, September 2022

[13] Qrisp website: https://www.qrisp.eu/, as of date 26.05.2023

[14] I. -D. Gheorghe-Pop, N. Tcholtchev, T. Ritter and M. Hauswirth, "Quantum DevOps: Towards Reliable and Applicable NISQ Quantum Computing," 2020 IEEE Globecom Workshops (GC Wkshps), Taipei, Taiwan, 2020, pp. 1-6, doi: 10.1109/GCWkshps50303.2020.9367411.

[15] OASEES project: https://oasees-project.eu/, as of date 26.05.2023

[16] EOSC marketplace: https://marketplace.eosc-portal.eu/, as of date 26.05.2023

[17] Raphael Seidel, Colin Kai-Uwe Becker, Sebastian Bock, Nikolay Tcholtchev, Ilie-Daniel Gheorghe-Pop, and Manfred Hauswirth, “Automatic generation of grover quantum oracles for arbitrary data structures”, Quantum Science and Technology, 8(2):025003, January 2023

[18] Qiskit contributors, “Qiskit: An Open-source Framework for Quantum Computing”, 2023, Zenodo, doi: 10.5281/zenodo.2573505

[19] Cirq Developers “Cirq (v1.1.0)”, 2022, Zenodo, doi: 10.5281/zenodo.7465577

[20] R. Seidel, N. Tcholtchev, S. Bock, C. K.-U. Becker, and M. Hauswirth, "Efficient Floating Point Arithmetic for Quantum Computers," in IEEE Access, vol. 10, pp. 72400-72415, 2022, doi: 10.1109/ACCESS.2022.3188251.

[21] DIN-Connect, Benchmarking Quantum Computers: https://www.fokus.fraunhofer.de/en/news/sqc/din_connect_challenge_2022_08, as of date 26.05.2023

[22] C. Kai-Uwe Becker, N. Tcholtchev, I. -D. Gheorghe-Pop, S. Bock, R. Seidel and M. Hauswirth, "Towards a Quantum Benchmark Suite with Standardized KPIs," 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C), 2022