# CHALLENGES OF USING QUANTUM COMPUTING TECHNOLOGY: A SOFTWARE DEVELOPERS PERSPECTIVE

*Luka Radujević*[1][ORCID 0000-1111-2222-3333], *Vladimir Mandić*[2][ORCID 0000-0001-6996-2222]
[1]*TIAC d.o.o., Novi Sad, Serbia*
[2]*University of Novi Sad, Faculty of Technical Sciences, Department of Industrial Systems and Management, Novi Sad, Serbia*

Abstract: *Quantum computing is a novel technology that operates on fundamentally different principles than classical computing. Despite their differing principles, both technologies share the common goal of creating solutions for specific problems that need to be tested and verified. The software development process has evolved and solidified over time within classical computing. However, the use of quantum computing technology with existing software development approaches is still an unresearched topic. This paper examines the challenges faced by software developers when working with quantum computing technology within the framework of the established software development process used in classical computing.*

Key words: quantum computing, software development process

## 1. INTRODUCTION

Quantum computing represents a groundbreaking technology that operates on principles fundamentally distinct from classical computing. Concepts such as entanglement and superposition enable problems that couldn't be solved using classical computing to become possible [2]. Some areas that could greatly benefit from this are physics, where complex simulations could be performed, as well as the world of finance, where numerous optimizations could be achieved. Due to the intensive development conducted by leading companies in the IT world, it is expected that quantum computing will start being applied in the software industry in the next few years [6].

Classical computing has a well-established software development paradigm [7]. The software development process in classical computing has evolved over time, guided by research and practical experience. Numerous works have investigated various aspects of classical software development, such as solution design, implementation, validation and testing, and deployment [4][9]. However, the integration of quantum computing technology within the current paradigms remains largely unexplored. The introduction of quantum computing technology also presents unique challenges to software developers.

To date, limited research has been conducted on the integration of quantum computing technology with the established software development process. While some works have touched upon specific aspects of quantum software development, such as algorithm design and optimization [3][11], a comprehensive analysis of the challenges faced by software developers remains a critical research gap. Understanding these challenges is vital for the advancement of quantum software engineering and the successful integration of quantum computing technology into real-world applications.

In order to advance the field of quantum software development, it is essential to understand the challenges faced by software developers when working with quantum computing technology within the context of the established software development process used in classical computing. The integration of quantum computing into the existing software development process necessitates the exploration of new techniques and methodologies to harness the power of quantum systems effectively.

This research objective is to address this research gap by examining the challenges faced by software developers when working with quantum computing technology within the established software development process used in classical computing. In order to address the objective, we used comparative research strategy for designing an empirical case study [10]. By identifying and analyzing these challenges, we can gain insights into the necessary adaptations and enhancements required for effective quantum software development. Additionally, we aim to provide recommendations for addressing these challenges and propose directions for future research in the field.

The rest of the paper is structured as follows. In section 2 we present background and research context, where main focus is on explaining concepts like quantum computing and quantum software engineering. Section 3 gives an overview of research method used in this study. In the Section 'Findings' we described case study we have made, and lessons learned from the case study. In the last section, we have discussed relation between findings in the related studies and the findings conducted in this study, as well as what future works should focus on.

## 2. BACKGROUND AND RESEARCH CONTEXT

The integration of quantum computing technology within the established software development process used in classical computing necessitates a thorough understanding of both the existing software development practices and the unique characteristics of quantum computing. In this section, we provide an overview of the quantum computing specifics as well as the potential benefits and applications of quantum computing in software development.

### 2.1 Quantum Computing

Quantum computing operates on principles fundamentally distinct from classical computing. Understanding the quantum software engineering is tightly coupled with understanding the quantum computing. As discussed in [1], currently most of the quantum languages are lower-level programming languages. Using those languages, programmers can directly manipulate low-level concepts e.g., quantum gates and circuits.
Survey [5] gives thorough overview of research related to quantum computing technology. Main building blocks of the quantum computers, qubits, are discussed in great details. Most important properties of qubits are superposition and entanglement, which give quantum computing power to overcome the power of classical computers in particular use cases [6]. However, it also introduces unique constraints, such as quantum noise, limited qubit coherence, and the need for quantum error correction.

### 2.2 Quantum Software Engineering

To date, the integration of quantum computing within the software development process has received limited attention in the literature. The concepts that are supposed to define the concept of quantum software engineering are presented in the paper [8]. The importance and need for quantum software engineering are emphasized by the fact that 100 researchers have signed the manifesto of quantum software engineering as a set of concepts on which quantum software engineering should be based.
In the paper [1], the authors discuss what is required to create quantum software that would be used in industry. They utilize an example of the conventional software development process, which includes the phases of requirement specification, design and architecture, development, testing, debugging, and maintenance. The result of the research conducted in the paper is the presentation of challenges that arise from quantum computing in certain phases of software.
There are couple of research discussing specific topics regarding quantum software engineering, such as testing and refactoring of the quantum code [3][11], but comprehensive analysis of the challenges faced by software developers during quantum software development process remains a unresearched topic.
The research presented in this paper aims to bridge the gap between classical and quantum software development processes. By investigating the challenges faced by software developers when working with quantum computing technology within the established software development process, we can provide insights into the necessary adaptations and enhancements required to effectively harness the power of quantum systems. This research also contributes to the broader field of quantum software engineering and serves as a foundation for future studies in this emerging area.

## 3. RESEARCH APPROACH

This research objective is addressing the challenges faced by software developers when working with quantum computing technology within the established software development process used in classical computing. In order to address the objective, we’ve been working on a quantum software development project. This project is specially designed to be able to be solved using both classical and quantum computing. We used comparative research strategy for designing an empirical case study. The case study consists of 3 steps which aimed to show all of the difficulties faced by experienced software developer in classical computing when attempting to solve a problem using quantum software development process. The steps followed in the examined case study were as follows:

1) Defining an abstract model of the software development process: software development process based on the well-known software development process in classical computing should be chosen and all of the components of that process have to be followed in the case study.
2) Defining the problem to be solved: in order to achieve comparative analysis, it was necessary to choose problem which could be solved using both classical and quantum computing.
3.) Applying the defined model of the development process during solution creation: In every step of the defined model, difficulties faced by software developer are noted and result of the analysis of all notes should be lessons learned.

Software development process we used in this case study consists of following components: problem defining, design and architecture, development, debugging and testing. Goal of the case study is addressing all of the difficulties faced by software developer in every single phase and try to summarize those difficulties in a couple of lessons learned.

## 4. FINDINGS

### 4.1 Solution implementation

Project described in the previous section is implemented by one of the authors of this research. As a first step, we needed to define an abstract model of the software development process which should be followed during the whole lifecycle of the project. Because all of the author's experience comes from the classical software development process, one type of the well-known software development process was chosen for this project. Defined development process consists of the following phases: (1) Requirements specification, (2) Solution design, (3) Solution implementation, (4) Debugging, and (5) Solution validation (testing).

Well-known phases like maintenance and deployment are intentionally excluded from this project because the aim of this research is to show experience of the software developer during the process of creating solution, not to investigate whole product's life cycle.

The next step which was intended to be done was defining the problem to be solved. It was important to choose a problem which could be solved using both classical and quantum technology, because only some kind of problems could be solved using quantum computing [3]. For this project, we chose the searching problem – problem of finding element in an unstructured collection of elements. When said unstructured collection, it means that collection of elements is not sorted or ordered in any way. Also, the type of data in the collection is not important, solution should be generic and data type should not be a limitation. Solution implemented by the developer is intended to be as performant as possible. The most important thing was that solution should be created using the defined software development process. Iterations over the phases are allowed.

As a first phase of the development process, specifying requirements in this project was not a big issue. The problem of finding element in the collection of data is one with most software developers are familiar with. The was no need for specifying subtasks or splitting the problem into smaller problems.

Phase of solution design started by making decision which programming language and platform to use. Because author's experience comes mostly from .NET and C# world, for implementing this project, Q# was chosen. The next step was an attempt of finding publicly available solutions for defined problem. The initial search only showed that there are a lot of knowledge gaps related to theory which should be additionally considered. Because current state of quantum software development requires programming on the low level, it was very challenging to be able to understand existing solutions and to use their full potential. A lot more time than expected is used for this phase. In the end, Grover's algorithm is chosen as the one to be implemented.

The phase of implementation started by exploring Q# and the quantum environment it comes with. Unlike various of ready for use solutions C# provides developers with, Q# is a low-level language which allows developers to have lot of control to the quantum gates and qubits itself. For somebody with classical software development background only, it was very hard to start being used to program hardware-level components. Implementing solution required numerous back iterations to the previous phase in order to understand theory and concepts in more detail. Also, this phase took much more time than expected in order to complete it.

The first phase of validating implemented solution is debugging. In the world of quantum software development, debugging is not a straightforward process as we know it in software development. Visual Studio, as an IDE used for implementing this project, allows us to put breakpoints and go through the code.

This feature helps developers up to some point, because reading the state in quantum computing means losing the information. This phase also required iterating back to the previous phases and reviewing the theory behind the actual implementation.
The last step in defined software development process is validating and testing of the solution. First approach for validating solution was manual process which included preparation of input values and checking if output is as expected. On the other side, automated testing in the solution is not achieved in this project. Besides the fact that Q# has a support for writing unit tests, it is very hard to test specific steps of the process when state of the system cannot be replicated and any interference with the state of the system could have an impact on the state itself.

## 4.2 Lessons learned

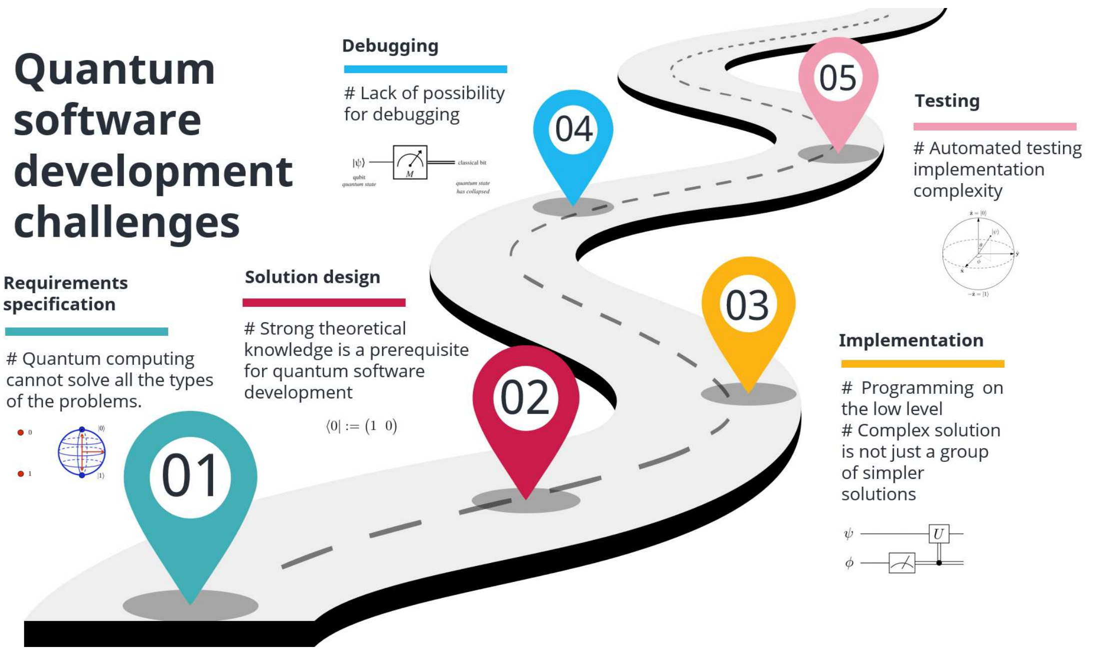
Quantum software development challenges
Requirements specification
# Quantum computing cannot solve all the types of the problems.
01
Solution design
# Strong theoretical knowledge is a prerequisite for quantum software development
⟨0| := (1 0)
02
Implementation
# Programming on the low level
# Complex solution is not just a group of simpler solutions
03
Debugging
# Lack of possibility for debugging
04
Testing
# Automated testing implementation complexity
05

*Figure 1: Quantum software development challenges infographic*

Working on the project which is based on quantum software development shows that there are lot of differences compared with the software development process in classical computing. In this section, difficulties faced during the development process are shown.
1. Quantum computing cannot solve all the types of the problems. When requirements of the software product are considered, it is highly important to consider could the problem being solved using quantum computing. Considering the fact that quantum software engineering and quantum computing are new technology, it is wort of trying to solve some problem using quantum computing, but developers should be aware that result could show lower performances compared to the one created using classical computing.
2. Programming on the low level. Using of the programming language Q# have been more difficult than using its classical equivalent C#. Programming in quantum computing requires strong knowledge of low-level programming, directly on quantum gates or qubits itself. Previous experience gathered from industry projects was not so useful in order to be able to solve problems in the quantum world.
3. Strong theoretical knowledge is a prerequisite for quantum software development. Designing solution in the classical computing requires investigation of possible solution and the previously accumulated knowledge is very helpful. The hardest part in that process is choosing the appropriate one between many of the offered solutions. All the designer needs in understanding of pros and cons of the solution. In the world of quantum computing, firstly, there are not so many available solutions. Designing the solution requires strong knowledge of quantum and quantum computing concepts. Experience on this project showed that there must be a proof made by physicists or mathematicians for some solution before it is used in quantum computing world. This phase consumes a big amount of time and it seems pretty

unnatural for software developer to understand so many theory concepts and particularly physics concepts in order to be able of making a solution.

4. Complex solution is not just a group of simpler solutions. In classical computing, developers are approaching complex problems by splitting it into a couple of smaller problem. Working on this project showed that solving a problem in quantum computing is not just combining well-known gates and qubits and getting the solution that way. It requires strong theoretical knowledge and theoretical proof in order to be applicable in quantum computing.

5. Lack of possibility for debugging. This challenge represents the hardest part of the whole process. Being able of putting breakpoint somewhere in the code or reading human-readable logs is somethings that really make programmers' life much easier. In the world of quantum computing, that is not possible because reading the value means losing the power of quantum computing. This part is something that creates very bad development experience.

6. Automated testing implementation complexity. Core concepts of the quantum computing, as superposition and entanglement are preventing developers of writing automated tests. When we tried to come up with solution to check some parts of the solution, it was not possible because of the fact that any measurement could damage the state and lead to the incorrect results.

## 5. CONCLUSIONS

Result of the conducted use case is a list of difficulties faced by software developers which tried to apply software development process model characteristic for classical computing in the quantum software development. Identified challenges have shown that there is a big difference between those two types of computing and that it is not easy for experienced software developer to start working on quantum computing project. There should be a lot of preparation before someone used to work in classical computing to be able to start working in quantum computing.

Although there are significant skills requirements that software developer needs to have in order to work on quantum project, experience from this project showed that development process established in classical computing was totally applicable to the quantum software development process. Phases were ordered in the appropriate and very natural way and helped software developer to create a product which should ultimately fulfill all the requirements specified at the beginning.

Results collected from this project confirm what is previously concluded by researchers in [2]. Authors have identified differences between classical and quantum software development process by phases of the development process. They have found that programming on the low level, deep understanding of quantum physics, as well as lack of possibility to debug code in standard way are something that is not characteristic for classical computing. Project described in this research confirms that those differences represent challenges for software developers and they should be aware of them before starting working on quantum software development environment.

Quantum software engineering is still not broadly applied in the software development industry, but once it has, this study should be reapplied. Evolution of quantum programming languages should be additional motivation to investigate how challenging it is for software developers to start working on the quantum software projects. Because quantum software engineering is at the beginning, this topic should be very interesting for many future studies.

## REFERENCES

Ali, Shaukat, Tao Yue, and Rui Abreu. (2022) *When software engineering meets quantum computing.* Communications of the ACM 65.4, 84-88.

De Stefano, Manuel, et al. (2022) *Software engineering for quantum programming: How far are we?* Journal of Systems and Software 190.

García de la Barrera, Antonio, et al. (2023) *Quantum software testing: State of the art.* Journal of Software: Evolution and Process 35.4.

Garousi, Vahid, and Mika V. Mäntylä. (2016) *When and what to automate in software testing? A multi-vocal literature review.* Information and Software Technology 76, 92-117.

Gyongyosi, Laszlo, and Sandor Imre. (2019) *A survey on quantum computing technology.* Computer Science Review 31, 51-71.

Hoefler, Torsten, Thomas Häner, and Matthias Troyer. (2023) *Disentangling hype from practicality: on realistically achieving quantum advantage.* Communications of the ACM 66.5, 82-87.

Kraeling, Mark, and Lindsley Tania. (2019) *Software Development Process*. Software Engineering for Embedded Systems. Newnes, 33-87.

Piattini, Mario, Guido Peterssen, and Ricardo Pérez-Castillo. (2021) *Quantum computing: A new software engineering golden age.* ACM SIGSOFT Software Engineering Notes 45.3, 12-14.

Shahin, M., M. Ali Babar, and Zhu L. (2017) *Continuous integration, Delivery and Deployment: A Systematic Review on Approaches, tools, Challenges and Practices*. IEEE Access 5, 3909-3943.

Wohlin, Claes, Martin Höst, and Kennet Henningsson. (2003) *Empirical research methods in software engineering.* Empirical methods and studies in software engineering: Experiences from ESERNET, 7-23.

Zhao, Jianjun. (2023) *On Refactoring Quantum Programs.* arXiv preprint arXiv:2306.10517.