A Novel Perception of Quantum Software: Theoretical, Engineering, and Application Aspects

Michael Felderer, Ricardo Pérez-Castillo, Mario Piattini, and Iaakov Exman

Abstract The chapter discusses the importance of quantum software and defines it as a multifaceted concept comprising a theoretical, engineering, and application viewpoint. Hence, it covers aspects of quantum software theory, quantum software systems, as well as quantum software laboratory. The chapter also outlines the entire book and its individual chapters structured into three parts on quantum software theory, quantum software system design, as well as quantum software laboratory and applications.

Keywords Quantum computing · Software engineering · Theory and experiments

M. Felderer
Institute of Software Technology, German Aerospace Center (DLR), Cologne, Germany

University of Innsbruck, Innsbruck, Austria

University of Cologne, Cologne, Germany
e-mail: Michael.Felderer@dlr.de

R. Pérez-Castillo (✉)
UCLM – University of Castilla-La Mancha, Talavera de la Reina, Spain
e-mail: Ricardo.PDelCastillo@uclm.es

M. Piattini
UCLM – University of Castilla-La Mancha, Ciudad Real, Spain
e-mail: Mario.Piattini@uclm.es

I. Exman
HIT – Holon Institute of Technology, Holon, Israel
e-mail: iaakov@hit.ac.il

I. Exman et al. (eds.), *Quantum Software*,
https://doi.org/10.1007/978-3-031-64136-7_1

1 Introduction

Quantum computing itself is a computational paradigm that explicitly uses properties of subatomic particles such as superposition, entanglement, and interference to achieve asymptotical speedups over classical algorithms on certain tasks [1]. It is gaining considerable attention from industry, academia, and public authorities alike and has been making great progress in recent years. In order to meet its expectations, not only hardware for quantum computing but also software is required. Quantum software is a key enabler for quantum computing and, as classical software, encompasses multiple facets.

As classical software, also quantum software is a multifaceted concept with several aspects. It covers algorithms, system software, application software, entire ecosystems, and hybrid systems, as well as suitable software and systems engineering concepts. In addition, new theoretical foundations for the concept of quantum software are required as well as concrete applications. This book aims to cover all these aspects and is therefore divided into three parts: Part I is on quantum software theory, Part II on quantum software system design, and Part III on the quantum software applications and laboratory.

The three parts of the book provide the reader with three different perspectives on quantum software. The first part takes the theoretical viewpoint into account. It emphasizes that one needs more than just informal software development models. The time has come to apply mathematical theories enabling convincing verification of the correctness of quantum software systems. The second part focuses on the software and system engineering viewpoint. Its goal is to show that in contrast to a dogmatic fixation on a single approach, one should take the variety of existing approaches into account to design novel quantum software systems. The third part of the book focuses on the laboratory and application perspective. Quantum software applications and laboratory experience are essential to avoid past mistakes and make quantum computing and its software a success.

The multiplicity of the sometimes conflicting perspectives offered by the different chapters of this book forces consideration of distinct factors involved in a problem, facilitating the path to the problem solution. The freedom afforded by the large variety of approaches is a blessing to be explored and not an impediment to rational decisions, as we try hard to demonstrate in this book. Moreover, it is an additional way to change how a quantum software system is perceived.

The following sections summarize the contributions of the individual chapters. Section 2 summarizes the two chapters on quantum software theory. Section 3 summarizes the six chapters on quantum software system design. Section 4 summarizes the four chapters on quantum software laboratory and applications.

2 Quantum Software Theory

Quantum computing requires theoretical considerations about the nature of quantum software. Quantum software adds additional aspects to the software engineering body of knowledge [2]. It is the result of a special combination of quantum and software. Two key aspects are modularity—separating modules, the meaningful subsystems of a system—and entanglement—linking modules when needed, in order to enable quantum software to run, e.g., in a simulation. Richard P. Feynman's visionary 1982 article entitled "Simulating Physics with Computers" [3] is a pioneer in the field of quantum computing. This is due to Feynman's inimitable style and his extensive analysis of the difficulties of quantum simulation of nature.

The second chapter, entitled "Simulating Quantum Software with Density Matrices: Reading Feynman on Fast-Forward," provides a novel reinterpretation of Feynman's paper [3] as a "quantum software" precursor. Feynman's proposal to represent the simulated system by a density matrix opens the way toward a mathematical quantum software systems theory. Density matrix modularization leads to *software modules* as high-level abstractions unifying conceptual software units, stimulating new software-related questions and novel quantum solutions. This chapter defines quantum software in terms of a conceptual software perspective and the density matrix as the rigorous bridge between concepts and qubits.

The third chapter, entitled "Superoperators for Quantum Software Engineering," reviews a superoperator-based approach to quantum dynamics. The approach is supposed to be concrete enough to be useful in quantum software and systems engineering, which necessitates gaining an understanding of quantum programming languages and possible approaches to equip them with formal semantics. The chapter discusses one particularly important superoperator-based formalism, i.e., linear superoperators acting on density operators. The chapter tailors the formalism toward software engineering research and indicates benefits in various application areas in that domain.

3 Quantum Software System Design

Based on theoretical considerations on quantum software and quantum computing, this part covers several contributions on quantum software system design. Quantum software is a multifaceted concept with several aspects. It covers algorithms, system software, application software, entire ecosystems, and hybrid systems, as well as suitable software and systems engineering concepts.

The fourth chapter, entitled "*QSandbox*: The Agile Quantum Software Sandbox," describes an *agile software sandbox* specifically designed for quantum software research and development. *QSandbox* is *itself modifiable* since its high-level modules are varied at will by quantum software developers. *QSandbox* has a series of unique features, to produce fast results when testing any recently modified

quantum circuit. It uses high-level abstraction meaningful modules, instead of low-level quantum gates of conventional simulators. It has instantly synchronized dual views—high-level quantum circuit and density matrix. In addition, it has uniform quantum and classical representation, implying the innovative idea of quantum circuits for classical software.

The fifth chapter, entitled "Verification and Validation of Quantum Software," focuses on classical software testing approaches for quantum software. For that purpose, 16 quantum software testing techniques, which have been proposed for the IBM quantum framework Qiskit, are gathered and illustrated based on a running example. The chapter concludes that researchers should focus on delivering artifacts that are usable without much hindrance to the rest of the community, and the development of quantum benchmarks should be a priority to facilitate reproducibility, replicability, and comparison between different testing techniques.

The sixth chapter, entitled "Quantum Software Quality Metrics," defines and empirically assesses a set of metrics for assessing the understandability of quantum circuits. The provided metrics fall into the categories circuit size, circuit density, single-qubit gates, multi-qubit gates, all gates in the circuit, oracles, measurement gates, as well as other metrics. Furthermore, a tool prototype called QMetrics is provided for automated calculation of the proposed metrics.

The seventh chapter, entitled "Quantum Software Ecosystem Design," presents scientific considerations essential for building a quantum software ecosystem that makes quantum computing available for scientific and industrial problem-solving. It is based on the concept of hardware–software codesign, which fosters a bidirectional feedback loop from the application layer at the top of the software stack down to the hardware. The approach starts with compilers and low-level software that are specifically designed to align with the unique specifications and constraints of the quantum processor. Then, the chapter presents algorithms developed with a clear understanding of underlying hardware and computational model features, and extends to applications that effectively leverage the capabilities to achieve a quantum advantage. The chapter analyzes the ecosystem from a conceptual view, focusing on theoretical foundations, and the technical view, addressing practical implementations around real quantum devices necessary for a functional ecosystem. It offers a guide to the essential concepts and practical strategies necessary for developing a scientifically grounded quantum software ecosystem.

The eighth chapter, entitled "Development and Deployment of Quantum Services," emphasizes that new techniques and tools are needed to facilitate access to quantum computing technology provided by cloud providers like IBM, Amazon, Microsoft, or Google. This helps developers to increase the level of abstraction at which they work with this technology. The chapter performs a technical comparison between different quantum computing service providers using a case study by performing empirical tests based on the Traveling Salesman Problem. The study highlights the differences between the major providers. In order to address these differences and reduce the vendor lock-in effect, the chapter makes three proposals: an extension of the Quantum API Gateway to support the different vendors; a code generator making use of a modification of the OpenAPI specification; and a

workflow to automate the continuous deployment of these services making use of GitHub Actions. This would allow programmers to deploy quantum code without specific knowledge of the major vendors, which would facilitate access and simplify the development of quantum applications.

The ninth chapter, entitled "Engineering Hybrid Software Systems," highlights that software modernization processes for transforming and migrating legacy software systems (which may include adding new existing quantum software) toward such hybrid software systems will be required. The chapter discusses the challenges of hybrid software and how software modernization (based on architecture-driven modernization) can be used as a reengineering solution for an effective evolution of classical and quantum software. This process makes it easier to combine both computing paradigms, quantum and classical. The modernization process consists of three phases, reverse engineering, restructuring, and forward engineering. The overall modernization process follows the Model-Driven Engineering (MDE) principles, and, therefore, it could be instantiated with different (meta)models. The main implication of the quantum software modernization process for practitioners is a set of challenges that may appear during the evolution of classical software systems toward hybrid software systems. Thus, software modernization helps companies to identify which components from their business models could be evolved, and how, or even to start new businesses following this new paradigm using techniques and standards which have been proved to be effective in solving such problems.

4 Quantum Software Laboratory and Applications

Based on the theoretical and software system design considerations, this part covers several contributions on quantum software laboratory and applications. It covers a concrete quantum computing technology, i.e., trapped-ion quantum computers, the application to quantum computer in the health domain, industrial application scenarios for quantum software engineering, as well as an empirical study to provide a comprehensive understanding of the current state of quantum software engineering.

The tenth chapter, entitled "Trapped-Ion Quantum Computing," presents trapped-ion quantum computing, which proves to be very suitable for the transition from tabletop, lab-based experiments to rack-mounted, on-premise systems that allow for operation in data center environments. However, several technical challenges need to be solved, and controlling many degrees of freedom needs to be optimized and automated before industrial applications can be successfully implemented on quantum computers situated within data centers. These necessary developments range from the architecture of an ion trap that fundamentally defines the supported instruction sets, over the control electronics and laser systems, which limit the quality of qubit operations, to the optimized compilation of quantum circuits based on qubit properties and gate fidelities. The chapter introduces the ion-trap quantum computing platform, presents the current technical state of the

art of Alpine Quantum Technologies GmbH (AQT's) ion-trapping hardware and rack-based quantum computing systems, and highlights parts of the execution stack.

The eleventh chapter, entitled "Quantum Software Engineering and Programming Applied to Personalized Pharmacogenomics," applies upcoming best practices of quantum software engineering to the development of a hybrid quantum/classical software system in the context of personalized pharmacogenomics. It reports on results from the QHealth project. The chapter concludes that in order to achieve quantum software that can really be used in health information systems, it is necessary to build it in an engineering way and without forgetting the good practices of software engineering. In fact, in the QHealth project, tools for design, quality, testing, estimation, and process management were proposed to implement the project.

The twelfth chapter, entitled "Challenges for Quantum Software Engineering: An Industrial Application Scenario Perspective," analyzes three paradigmatic application scenarios for quantum software engineering from an industrial perspective. The use cases cover (1) optimization and quantum cloud services, (2) quantum simulation, and (3) embedded quantum computing. From the use case analysis, the chapter concludes that quantum programming today mostly means custom-tailoring a quantum solution to a very specific instruction set of a specifically developed special-purpose quantum computer. The tendency in software engineering today is, however, to abstract exactly from those low-level details and instead focus on requirements and design issues. Hence, recently outdated, former core disciplines of mainstream software engineering research like compiler construction and instruction set architecture design will become highly relevant again.

The thirteenth chapter, entitled "Quantum Software Engineering Issues and Challenges: Insights from Practitioners," presents an empirical study based on a survey and expert interviews. Its aim is to provide a comprehensive understanding of the current state of quantum software engineering. Results show that there is great enthusiasm and interest in quantum programming, with abundant educational and experimental repositories indicating a fertile ground for innovation. The potential applications of quantum computing, especially in fields like chemistry, physics, and cryptography, are promising, and this has led to a growing community of developers and researchers eager to explore and contribute to this emerging field. However, many challenges must be overcome before the full potential of quantum software engineering can be realized. These challenges include a steep learning curve, a lack of standardized frameworks, hardware limitations, and a nascent stage of community collaboration.

5 Conclusion and Acknowledgment

This chapter has discussed the importance of quantum software. It covers aspects of quantum software theory, quantum software systems, as well as quantum software laboratory. The editors of this book want to thank all chapter authors for their

valuable contributions. Furthermore, the editors want to express their gratitude to all participants of the First Working Seminar on Quantum Software Engineering (WSQSE 22) [4] for the fruitful discussions. WSQSE 22 was held on December 15–16, 2022, in Innsbruck, and during that seminar, the idea for this book was born from the fruitful discussion.

References

1. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press (2010)
2. SWEBOK V3.0. https://www.computer.org/education/bodies-of-knowledge/software-engineering/topics
3. Feynman, R.P.: simulating physics with computers. Int. J. Theor. Phys. **21**, 467 (1982)
4. Felderer, M., Taibi, D., Palomba, F., Epping, M., Lochau, M., Weder, B.: Software engineering challenges for quantum computing: Report from the First Working Seminar on Quantum Software Engineering (WSQSE 22). ACM SIGSOFT Softw. Eng. Notes. **48**(2), 29–32 (2023)