Challenges for Quantum Software Engineering: An Industrial Application Scenario Perspective

Cecilia Carbonelli, Michael Felderer, Matthias Jung, Elisabeth Lobe, Malte Lochau, Sebastian Luber, Wolfgang Mauerer, Rudolf Ramler, Ina Schaefer, and Christoph Schroth

Abstract Quantum software is becoming a key enabler for applying quantum computing to industrial use cases. This poses challenges to quantum software

C. Carbonelli · S. Luber
Infineon Technologies AG, Neubiberg, Germany
e-mail: cecilia.carbonelli@infineon.com; sebastian.luber@infineon.com

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

M. Jung
University of Würzburg, Würzburg, Germany
e-mail: m.jung@uni-wuerzburg.de

E. Lobe
Institute of Software Technology, German Aerospace Center (DLR), Brunswick, Germany
e-mail: elisabeth.lobe@dlr.de

M. Lochau (✉)
University of Siegen, Siegen, Germany
e-mail: malte.lochau@uni-siegen.de

W. Mauerer
Technical University of Applied Sciences/Siemens AG, Regensburg, Germany
e-mail: wolfgang.mauerer@othr.de

R. Ramler
Software Competence Center Hagenberg, Hagenberg, Austria
e-mail: rudolf.ramler@scch.at

I. Schaefer
Karlsruhe Institute of Technology, Karlsruhe, Germany
e-mail: ina.schaefer@kit.edu

C. Schroth
Fraunhofer IESE, Kaiserslautern, Germany
e-mail: christof.schroth@iese.fraunhofer.de

I. Exman et al. (eds.), *Quantum Software*,
https://doi.org/10.1007/978-3-031-64136-7_12

engineering in providing efficient and effective means to develop such software. Eventually, this must be reliably achieved in time, on budget, and in quality, using sound and well-principled engineering approaches. Given that quantum computers are based on fundamentally different principles than classical machines, this raises the question if, how, and to what extent established techniques for systematically engineering software need to be adapted. In this chapter, we analyze three paradigmatic application scenarios for quantum software engineering from an industrial perspective. The respective use cases center around (1) optimization and quantum cloud services, (2) quantum simulation, and (3) embedded quantum computing. Our aim is to provide a concise overview of the current and future applications of quantum computing in diverse industrial settings. We derive presumed challenges for quantum software engineering and thus provide research directions for this emerging field.

Keywords Quantum computing · Software engineering · Quantum software engineering · Industrial use cases · Software development

1 Introduction

Quantum computers (QCs) are a reality today, but quantum software development is in its very infancy. Although many small-/medium-sized quantum programs have been written over the years to demonstrate the potentials of quantum computing, barely any of these examples can be seriously called *quantum software*. In other words, there is no such thing as quantum software to date [28].

In this regard, *software engineering* (SE) is concerned with supporting and improving the development, application, and maintenance of software-intensive systems [92]. SE employs scientific methods, business principles, structured process models, and predefined quality goals to cope with the complexity of software as a whole. Current mainstream SE research for classical (i.e., non-quantum) software comprises design principles (e.g., high-level modeling languages fostering abstraction and modularity), development practices (e.g., tasks, roles, and responsibilities), and tool support (e.g., Integrated Development Environment (IDEs), code generation, static analysis, version control, issue tracking, unit testing, debugging, etc.). This perspective of SE research on software development is, however, mismatching the current status of quantum software. Zhao et al. were some of the first to coin the term *quantum software engineering* (QSE) to summarize any effort to adopt established SE principles and practices to make them also work for quantum software [111]. However, in this chapter, we take on a contrary perspective: research on QSE should, as a first step, identify, understand, and tackle short-term engineering challenges for better support of, usually fully manually crafted, small-/medium-scale quantum programs today (i.e., focusing on the programming and deployment phases). More sophisticated and mature concepts including high-level software abstraction as propagated, for instance, in the context of requirements elicitation, object-oriented design patterns, software maintenance and evolution, and re-engineering are out of scope for now due to the lack of any accessible examples and use cases. To meet the short-term goals, quantum SE should first of all focus on the following challenges.

- Make quantum computing accessible to developers and users through appropriate processes, methods, and tools.
- Facilitate hybrid quantum computing through a combination of classical SE and QSE concepts based on a generic description of a computational problem and (quantum) platform constraints.
- Provide benchmarks and benchmarking processes, methods, and tools for assessing quantum advantage as well as constraints that arise from the integration of quantum software components in an overall (hybrid) software system.

Our goal is to assess the short-term requirements and challenges of SE in the upcoming era of quantum computing. These requirements and challenges are already relevant to the noisy intermediate-scale quantum (NISQ) era. In contrast to other recent works on this subject [111, 6, 101, 110], we do not follow a top-down approach, but instead, illustrate the status quo of QSE by considering a selection of industrial application scenarios. For each application scenario, we first provide a short general description and then describe selected recent use cases to characterize the common aspects of the respective scenarios. Based on these descriptions, we derive in a bottom-up manner the key challenges for QSE with respect to these application scenarios. Our goal is to gain a better understanding of the principles and practices that will most likely support the development of software systems that solve problems that, at least partly, involve quantum computations. Our claim is that, from an SE point of view, quantum computation is not a new programming paradigm in the first place, but, first of all, a new *computational architecture*. The novel conceptual thinking required for effectively exploiting the frequently promised *quantum advantage* is crosscutting all classical development phases and hierarchies of software systems. Quantum computing will thus potentially influence SE as a whole as we know it today [89]. Nevertheless, we argue that established solutions developed in SE research over the past decades will not all suddenly become inappropriate and obsolete due to the advent of quantum computing, but instead require careful rethinking and adjustments to also cope with the key characteristics of quantum software. Many of these characteristics and possible side effects apparent in quantum computations have been considered before in other contexts, whereas the inherent pervasiveness of these characteristics in a quantum setting is indeed a novel aspect. These characteristics include, for instance, the probabilistic nature of computational outcomes and the lack of reference architectures (although `Qiskit` may be seen as a de facto standard today for the majority of computational approaches).

"While many of quantum computing's promised capabilities could be revolutionary, the realization of this promise requires breakthroughs in several areas, including improvements in the quality of qubits, error correction, and a demonstrable set of practical applications" [28]. The inflated expectations may result in a *quantum winter* similar to what we experienced with AI, where it took a long time to turn promising theoretical concepts into reality. Thus, the immediately necessary contributions of the SE community to advancing quantum computing lie in moving from first demonstrable examples to real-world applications with practical impact.

2 Paradigmatic Application Scenarios

We next describe potential application domains of QSE by means of paradigmatic application scenarios as illustrated in Fig. 1:

- **Application Scenario 1.** Provide quantum computing capabilities as a cloud service to solve optimization problems or machine learning tasks (quantum-computing-as-a-service).
- **Application Scenario 2.** Perform physical simulations with quantum programs developed by domain experts in a machine-oriented low-level manner.
- **Application Scenario 3.** Embed quantum processing units (QPUs) as integrated components into hybrid safety- or mission-critical software systems with a special focus on nonfunctional properties.

The selection of these application scenarios is driven by industrial and academic experiences of the authors and is aligned with the core use cases of the QUTAC Consortium [11]. Our aim is to illustrate the diversity of application domains and different perspectives on quantum computing, ranging from recent black-box and white-box views to embedded quantum computing.

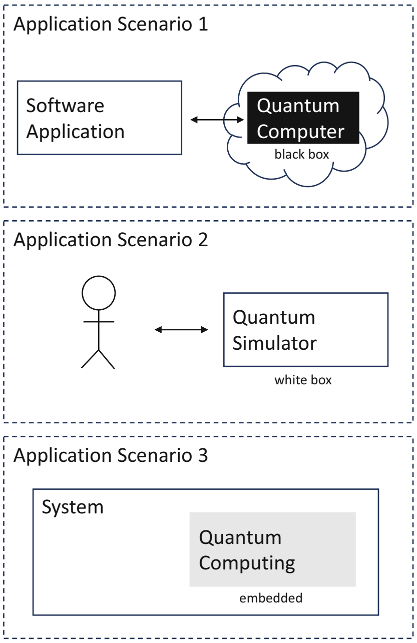

Fig. 1 Application scenarios for quantum computing

2.1 Application Scenario 1: Quantum Cloud Services

2.1.1 Use Cases and Examples

Quantum computing brings new opportunities for **solving optimization problems**, which are among the first industrial applications of the technology [11]. One example is the *flight-gate assignment (FGA)* problem in airport and air traffic planning, where the assignment of incoming flights to gates shall be optimized to minimize transfer times [94, 81]. This scheduling problem belongs to a class of NP-hard combinatorial optimization problems. Further examples include *Electronic Design Automation* (EDA) such as placement and routing on design chips and fault detection in electrical power networks [68], trajectory optimization in air traffic management [95], paint-shop scheduling [108], and planning problems in highly individualized mass production [9].

NP-hardness implies that, in practice, real-world instances can only be solved by approximation algorithms or heuristics. Here, quantum computers, taking advantage of entanglement, superposition, and interference, could potentially speed up and improve the optimization. One key property of the mentioned problems is that they can be solved **offline**: one problem instance is solved once, usually without critical time constraints, and the result is used to do something substantial, either conducting further research or going in an operational state, for instance, by applying the optimized flight schedule in an airport scenario.

2.1.2 Approaches and Challenges

A possible solution to bypass insufficient local computational power for effectively solving hard optimization problems is to **pass the work to a quantum cloud system**. For instance, D-Wave's Leap service [23] provides connections to quantum annealers or IBM's Qiskit interfacing to their quantum machines [71].

It has been argued that handling the offloading of such computations does not pose any new challenges to SE [50] as quantum computing essentially appears as a black box with well-defined interfaces. However, an open issue is to properly design such interfaces and to **formulate optimization problems** being tractable by quantum processing units (QPUs). First, an emerging optimization problem may be identified as computational bottlenecks within larger application contexts. These problems are either spotted by mathematical analysis during the design phase or during an optimization stage using a profiler; ideally, they match one of the known quantum primitives [39]. To this end, developers have to **refactor the overall software system** to isolate and replace the computational component by calls to quantum cloud services. Yet, there may be many such components that are closely tied to specific requirements of the overall system and which are the result of decades' worth of research and optimization [24]. This makes replacing them a nontrivial endeavor. Examples include subtasks of database management systems like join ordering [84, 85], multi-query optimization [98] or transaction scheduling [15].

Using quantum computing to speed up tasks previously solved by components designed and optimized for classical computers thus requires careful analysis. This includes understanding the underlying problem as well as exploring possible quantum speedups under varying workloads, input data characteristics, etc., while simultaneously retaining crucial, yet unrelated functional and nonfunctional properties of the overall system. Established SE techniques and tools (e.g., for performance analysis and refactoring) may help.

However, it is fair to say that the understanding of **what benefits quantum computers can provide** for what specific problems **is far from being well understood** in comparison to the state of the art in classical algorithms, and *also* in terms of system architecture.

While the potential speedups of seminal approaches [64] like Shor's algorithm (and, more generally, quantum phase estimation) or Grover search are rigorously established, the impact of imperfections on these algorithms makes any practical application considerations quickly prohibitive [43]. Likewise, even the actual requirements on the hardware of future machines for comparatively simple co-variables like the number of qubits is subject to discussion, and depends not only on many low-level details of the underlying hardware, but also on the actual payload algorithms [78]. A substantial body of the existing literature is dedicated to establishing a comprehensive understanding of the theoretical advantages of quantum sampling approaches [44]. Yet, concrete applications of these techniques are thinly spread, and their practical gain especially in comparison to existing heuristics and approximations is still under initial exploration [27].

The situation becomes even less straightforward for the more recent class of variational quantum algorithms and quantum approximate optimization algorithm (QAOA)-style approaches [17, 13]. While it is known that an efficient simulation of specific variants of QAOA would have strikingly unattractive and unexpected consequences for some of the pillars of computational complexity theory, entirely classical replacements for other variants are also known [61]. Likewise, the understanding of how to construct efficient classical surrogates for variational algorithms has considerably increased recently [87, 86], and restricts potential quantum advantage to increasingly narrower domains. When—unavoidable—practical constraints are taken into account [105, 79], determining a fair basis for comparison is a not yet satisfactorily resolved problem [12, 46], even ignoring the substantial limitations of currently available hardware.

A major challenge is to identify, factor out, and transform optimization subtasks suitable for quantum computers or annealers and their specific computing architectures [25, 48]. Several transformation steps usually reformat the optimization problem. In addition, "glue" code to connect classic and quantum parts is required. Tools like `Quark` [55][1] enable users to easily formulate and transform optimization

[1] See also the list of contributors (link in PDF).

problems, and to handle experiment dispatch and analysis. Likewise, approaches for recommending solution strategies for optimization problems using quantum approaches have been suggested [70]. However, such tools to support interacting and experimenting with quantum computers are still subject to research [103].

2.1.3 Conclusions

We are in the phase of evaluating the potentials of quantum computing in solving optimization problems. Providing such capabilities as reliable (black-box) services, however, requires an improved understanding of machine properties obtained from experiments and benchmarking. This necessitates many iterations of interactions with the quantum hardware for parameter tuning. Software development efforts, therefore, increase significantly when dealing with quantum hardware in contrast to well-established classical approaches as this fine-tuning requires not only software skills but also deeper knowledge in fundamental quantum physics. We assume this up-front investment will eventually pay off: if a fast heuristic solution is available and easy to access, a user will simply call quantum optimizers as a black-box cloud service over a well-defined interface, hiding transformation complexity and specific hardware requirements.

Summary Quantum cloud services will allow for accelerating mathematical optimization problems. Automatic means of transforming existing formulations into quantum descriptions have become available; yet, it remains a software architecture and engineering challenge to identify appropriate problems. Integrating quantum solvers into applications from a black-box perspective, including interface design, remains a minor SE challenge. However, the underlying quantum computing software stack, including the compilation and hybrid computing process, requires new quantum software engineering approaches.

2.2 Application Scenario 2: Quantum Simulation

2.2.1 Use Cases and Examples

Quantum simulation is one of the most promising application scenarios for quantum computing. It can help in understanding real-world chemistry and physics phenomena, improving design methodologies and making experiments much more effective. Simulating quantum mechanics on classical computers is a hard computational problem,[2] and determining relevant properties of quantum systems, e.g., finding their minimal energy, is even harder. To efficiently simulate a quantum system, the simulator might rely on quantum-mechanical dynamics. The basic idea of a quantum simulator is to use a controllable quantum platform to replicate dynamic or static properties of another, usually less controllable, quantum system [51]. This is similar to using wind tunnels for testing aerodynamic properties of reduced-scale models in a controlled environment, and then to transfer gained information to full-scale objects in the (uncontrolled) real world. With the rapid growth of quantum computing capabilities, the interest in **(quantum) material science** has also risen significantly. This field targets a large variety of applications ranging from the *design of more efficient batteries and catalysts* to the study of *innovative sensing materials for consumer and automotive applications* [66].

For the latter task, many candidate compounds have to be efficiently screened and evaluated to select or design the best materials with respect to the desired properties. This implies large effort and costs in terms of material procurement, measurement equipment, and setup. Direct simulations of the material properties could drastically reduce the required resources and significantly accelerate the discovery process reducing time-to-market. Here, quantum systems promise a fast and more precise simulation tool of real-world mechanisms than their conventional counterparts.

Likewise, the study of new storage materials and the development of innovative battery technology is being pushed by several emerging and established applications, ranging from electric and light electric vehicles to solar energy storage systems and robotics. Researchers aim at understanding the mechanisms impacting efficiency, stability, and faster charging of battery operations to predict real-world performance. Yet the first fundamental step, again, remains the selection of apt chemical compounds. New families of disruptive active materials such as Lithium-Ion (Li-ion) and Lithium-Sulfur (Li-S) offer four times higher energy density than Li-ion batteries. From a modeling perspective, it is crucial to describe the solid electrolyte interphase forming on the battery anode and to define its durability and long-term performance. Classic DFT, multi-physics simulations, and measurements have not provided satisfactory answers particularly in terms of accuracy . Quantum computing can offer a closer characterization of the key chemical properties of battery cells such as equilibrium cell voltages, ionic mobility, and thermal stability.

[2] Problems efficiently solvable by QC belong to complexity class bounded-error quantum polynomial (BQP), the quantum analog of BPP. The relation between BQP and classical classes like NP poses many open questions. The **dynamics** of a quantum system (compute output of a quantum circuit given an initial state) is BQP-hard [36], which makes it likely intractable for classical computers, but doable for quantum machines for a class of natural Hamiltonians in BPQ. Inferring *global* properties of a quantum system (given a quantum circuit, is there a state that produces a desired output? What is the minimum energy eigenstate for a given Hamiltonian?) belong into QMA, a probabilistic quantum analog of NP [1], and is intractable even for quantum computers . It is even possible to give physical problems that are undecidable, at least within the limit of infinite size [22]. Quantum SE needs to be aware of such peculiarities to properly ascertain the feasibility of architectures and designs by avoiding illusory, inflated expectations of potential gains.

The quantum simulation often boils down to obtaining the ground state energies of various molecules of increasing complexity [26]; likewise, physical characteristics like dipole moments have also been calculated [77].

2.2.2 Approaches and Challenges

Programmable universal quantum computers can simulate quantum mechanical processes [18, 40, 7, 54, 16]. Such simulations are specified using software (e.g., using **domain-specific languages**), which takes this topic into the focus of SE. However, different approaches to quantum simulation (analog simulation, digital simulation, combinations thereof, and hybrid quantum-classical algorithms) differ in their implications. In each case, and in contrast to other forms of quantum computation, quantum simulation requires **awareness of the Hamiltonian** underlying the task (the Hamilton operator (or *Hamiltonian*) of a system is, roughly speaking, a mathematical object[3] that provides information about a physical system. It is closely related to the energy spectrum,[4] and governs time-evolution of a quantum system. The Schrödinger equation combines Hamiltonian and quantum states, which are mathematically described by the wave function, into a differential equation).

Analog quantum simulators [18, 20] are physical systems that mimic other quantum systems (or a class of models) by closely reproducing the system's characteristics. Hence, their Hamiltonian should be as similar as possible to the simulated system. *Digital quantum simulation* is based on decomposing the Hamiltonian into operations implementable in the simulator by single- and two-qubit gate operations. This is more flexible than analog quantum simulators and enables us to overcome the limitations of the simulator system itself. Furthermore, it allows for quantum error correction and universality in a *"fully universal" quantum*

[3] We have been deliberately careful to avoid confusing the physical concept of a dynamical observable that can be measured with the mathematical operator/object to which it corresponds in the formal description.

[4] Many textbooks on quantum mechanics simply state that the Hamiltonian represents the total energy of a system, sometimes requiring this as a fundamental postulate. There are reasons to avoid such strong statements, both from a fundamental perspective (in the canonical approach of replacing physical quantities in the Hamilton function H of classical mechanics with operator-valued quantities, H is always conserved, but does, as Legendre transform of the Langrangian, not automatically equate to the sum of potential and kinetic energy; the approach to deriving a quantum Hamiltonian from energy-momentum relations delivers different results for the non-relativistic and relativistic case; and approaches based on space-time symmetries need to introduce empirical factors that relate the quantum Hamiltonian to classical energies), and from a practical point of view that concerns the software engineering aspects of quantum simulations. It is fairly common in this field to work with *effective Hamiltonians* that describe only degrees of freedom relevant for a particular task (for instance, Spin Hamiltonians in spectroscopy, the Ligand Field Hamiltonian of coordination chemistry, or the Hückel Hamiltonian for aromatic systems, which all carry a certain relevance for quantum chemistry), and therefore do not deliver a complete energy spectrum. Correctness checks, invariants, and the interpretation of results must adapt to such circumstances, and require awareness from the software side.

computer. If the simulator offers a universal set of perfect quantum gates, then the model can simulate a wide class of Hamiltonians [54], albeit the computing effort may vary depending on the types of gates. Some implementation technologies for QCs in use today are particularly well suited for quantum simulators. An example is Rydberg atom arrays [63, 102] that provide identical and long-lived qubits with strong coherent interactions. To represent the physical properties of the simulated system, the properties of the simulator correspond well to these, especially when analog simulation steps are involved. At least for this aspect, this challenges the idea that abstraction layers [10], despite proven useful classically, can satisfactorily eliminate differences between implementation platforms.

Industrial experience with quantum simulation problems gained by some of the authors shows that the exact boundary between digital simulation and optimization is not always clear. Especially quantum-classical hybrid algorithms—most importantly, the variational quantum eigensolver (VQE) [97]—rely on optimization methods to determine observables like the ground state energy of molecules based on a physical model. It is hypothesized that VQE, which at its core is independent of the simulated problem, will provide improved modeling accuracy over classical approaches like DFT. However, engineering challenges remain such as hardware-dependent **noise compensation**, an understanding of the differences between the many available variants of VQE [34] (requiring **problem-dependent benchmarking** [76]), and **determining optimal quantum-classical splits**. Especially the latter topics fall within the responsibilities of SE, but it might also be possible to improve noise handling based on software-centric methods. Also, the **depth reduction of circuits** generated from Hamiltonian descriptions is an important goal, in which compilers may play a crucial role (see, e.g., [30, 49, 82]). As with other use cases, resource usage and scalability in general need to be addressed by QSE.

Despite initial steps taken on problems of industrial scale, explorations are still in an early phase with already important collaborations emerging between large chemical and computing technology corporations [19]. Currently, the effort of finding appropriate Hamiltonian models by far exceeds the software implementation effort; knowledge of physical principles and details by far outranks the challenges of transcribing these into the quantum framework. While the modeling task in the classical domain is routinely reduced to a well-informed parametrization of canned DFT software, quantum tools—even given existing frameworks support [71]—require high manual programming effort.

2.2.3 Conclusions

SE tasks in quantum simulation include algorithm selection, determining the influence of mathematical/physical details on nonfunctional and functional properties, and comparing quantum and hybrid architectures to classical approaches and heuristics. Many revolve, in a broader sense, around the topic of **testing**. As the goal of quantum simulation is to exceed the computational capabilities of classical approaches, this opens up new research challenges. Testing quantum simulations comprises ensuring (a) model correctness and (b) correctness of circuits generated from the model. After establishing a Hamiltonian description of the system, empirical measurements on the actual physical system can be performed and compared to the simulation results. The resulting circuit generator is then trusted, and quantum simulation based on the generators can be used to explore the properties of novel, previously unexplored materials.

From an SE point of view, recent attempts lift established techniques for end-to-end testing of classical computations to components with probabilistic behavior [62, 41, 35, 38]. This includes novel notions of testing oracles based on distance measures for execution trace distributions and statistical criteria for approximating error probability by the number of repetitions of test runs. More involved quantum phenomena like superposition and entanglement of computational states are not yet properly addressed by these approaches. This, first of all, requires new abstractions concerning the notion of *observations* in testing reflecting the destructive nature of quantum measurements which obstructs established testing practices like interactive debugging [62].

Further properties of quantum states and circuits are also not suited to established testing methods: as there are usually no classical control branches in quantum circuits, structural code coverage criteria are not applicable, which renders well-established, elementary software testing concepts [92] useless. Likewise, localization of faults is unlike harder for quantum circuits than for classical programs, given that entangled states can intertwine arbitrary parts of a circuit and mutually influence each other . Not just the stochastic nature of quantum measurements but also the impact of imperfection and noise in quantum circuits obstruct the definition of proper test oracles. Here, we need to distinguish unavoidable variations caused by quantum measurements from variations due to (classically) probabilistic algorithmic elements from variations induced by noise and imperfection. Distinguishing between such different probability distributions is no new challenge, but there are quantum specifics: for instance, the amount of information to be recorded for a meaningful statement (e.g., by estimating the required number of samples for a desired precision and bounded error probability via Hoeffding's inequality [65], or randomized measurement procedures [31] that estimate quantum properties from classical observations) requires future research in QSE. Well-principled guidelines can eliminate the need for individual software engineers to be aware of such statistical peculiarities.

Other verification approaches for quantum simulation include up-front correctness validation of models (e.g., finding physical invariants that can be probed with accessible measurements), equipping a model's software representation (or the representation of the simulation approach) with a formal semantics honors quantum aspects (e.g., [58, 14, 21, 32]) that allows us to verify specific properties and correctness of generated modeling circuits by decomposition techniques (see, e.g., [67, 99]).

Summary Quantum simulation can benefit from established means of SE to formulate and describe models of physical systems whose properties can be simulated on quantum computers. Efforts evolve more around a physical understanding of the employed models rather than programming. Validation and verification techniques, as well as architectural decomposition into quantum and classical aspects, will rely on established, yet to be adapted SE approaches.

2.3 Application Scenario 3: Embedded Quantum Computing

2.3.1 Use Cases and Examples

Embedded software systems are purpose-built for specific tasks. In contrast to general purpose and high-performance computing systems (Application Scenario 1), embedded systems operate under restricted resources, on specific hardware platforms, and have to meet distinct quality requirements like **real-time constraints or safety guarantees**. Safety measures prevent material damage and harm to individuals and deeply influence hardware and software co-design of classical embedded software [57].

2.3.2 Approaches and Challenges

We recently observed a convergence between embedded systems and high-performance computing [42], for instance, in autonomous driving, avionics, and control systems. We expect embedded systems to require even more computational resources in future applications. Hence, quantum computing may also play an important role in **hybrid embedded scenarios** by utilizing **quantum accelerators** for solving particular computational tasks [105]. To the best of our knowledge, no approaches have been investigated so far to facilitate quality assurance techniques and tools for embedded quantum computing. Meeting these requirements in QPU accelerated hybrid systems is complicated by the dominance of iterative, probabilistic algorithms; yet, since almost all known quantum algorithms that operate on perfect error-corrected quantum systems are also inherently *probabilistic* [56], the problem will also extend after the NISQ area. Open research questions include how to improve understanding of termination properties and convergence toward sufficiently accurate results in iterative algorithms [33, 3], as well as the role of classical optimization components [114, 96] and result degradation. But, perhaps counter-intuitively, also possible improvements [53] by imperfections and noise [4, 100] are important research questions.

In many application domains, embedded co-design development processes must achieve (safety) certifications. It is an open question how established approaches can be adapted to quantum computing including entirely novel qualification approaches aligned with QPU peculiarities. Therefore, we may expect that system engineering will play a larger role in hybrid embedded quantum computing than for classical applications.

Prior work in safety-critical embedded systems deals with probabilistic algorithms and machine learning (e.g., neural networks) in the context of unreliable hardware. Measures include redundant computation, error correction [93], as well as more high-level concepts like safety cages [45] and static partitioning [73], *Digital Dependability Identities* (DDI) [74], and *Dynamic Risk Management* (DRM) [75]. It is not obvious if and how these approaches can be adopted for quantum computing. It is also crucial to consider how to integrate QPUs into existing embedded development processes and infrastructures. This includes interface design (at the physical and protocol level) to ensure proper timing and co-scheduling of computational tasks offloaded to a quantum component. The integration of QPUs further impacts the software operating systems level and middleware layers. Given the strong influence of imperfection of QPUs in the near and midterm [13], QPU integration will also impact co-design of hardware and algorithms to ensure computational advantages for a given set of problems. The established approaches to hardware–software co-design are currently adapted to interactions between QPUs and classical system components [52, 5], with efforts ranging from traditional embedded systems design to integration with high-performance computing [105, 88], all of which also pose software engineering challenges. The feasibility of co-design decisions strongly depends on the underlying physical implementation technology, which influences the quality properties of any software executed on top.

Since embedded systems are employed in industrial and cost-sensitive domains, economic considerations are also important in a quantum setting, especially given that even in the upcoming era of fully error-corrected quantum computers (but even more so in the NISQ era), different physical implementations of the computational concepts will offer different characteristics depending on their physical implementation [104]. A quantum approach with marginal improvement over existing solutions at the expense of inflating the bill of materials (or other development costs) is neither intellectually satisfying nor economically desirable. Embedded quantum SE must consider these issues.

2.3.3 Conclusions

The main challenges to enable hybrid embedded quantum computing include novel co-design principles and practices to adopt quality assurance techniques (e.g., embedded systems testing) and corresponding certification processes to a quantum setting. This is particularly crucial in safety-critical application domains. In the near term, we may expect quantum computing to find its way into large-scale embedded systems only (e.g., in CT scanners). In contrast, the physical size of recent quantum computers is the main limiting factor for small-scale, mobile use cases such as automotive *Electronic Compute Units* (ECU). These limitations of first-generation QPU are not quantum inherent, and future quantum technology may provide quantum accelerators fitting into small, well-integrated embedded systems.

Summary We expect that QPUs, given increasing miniaturization, will be deployed as accelerators in embedded use cases. This requires applications (and extensions) of established co-design methods from embedded SE that also lean substantially toward systems engineering. Quality assurance, certification requirements, and economic and physical constraints will play pronounced roles.

3 Promises and Perils of Quantum Software Engineering

3.1 Promises and Opportunities

Application scenario 1 is aligned with classical SE for developing complete software solutions by making use of quantum cloud services, whereas application scenario 2 crosscuts classical SE and instead seeks support of craftsmanship by individual experts. Application scenario 3 demands principles and practices similar to systems engineering for quality-aware integration of heterogeneous software/hardware components on a computational platform. From these observations, we conclude that the work with quantum computing is, and will be, similar to the development process using embedded accelerators, such as GPUs or special-purpose hardware (see Fig. 2). Similar to hardware–software co-design approaches, we expect that hardware–software–QC co-design processes will be required to split classical from quantum software parts [29, 69]. Likewise, a number of proposals have been made regarding more general questions of software architecture for quantum-classical hybrid systems, for instance [90, 80, 37].

After the diverse software parts are completed and tested as separate units (taking into account that quantum aspects bring additional challenges to reproducibility aspects [60]), an integration test step is required. Ideally, those steps will be embedded into continuous engineering processes [8], e.g., by making use of virtual hardware platforms or simulators for faster feedback cycles. We next discuss challenges of QSE by considering the respective SE phases. While many of these challenges have already been mentioned in recent surveys on QSE [111], our attempt is to relate these aspects to the insights gained from all three application scenarios described above.

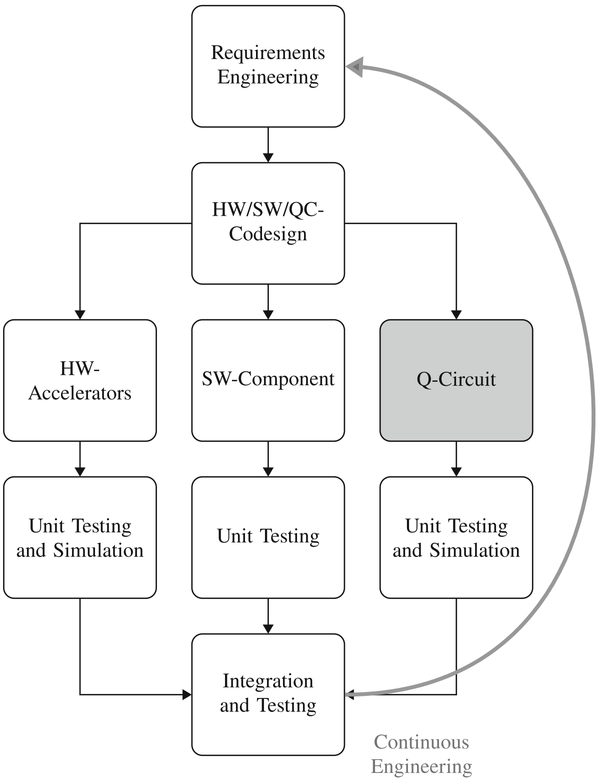

Fig. 2 Quantum software development process

Requirements Engineering The requirements engineering phase will not fundamentally change as requirements, by definition, deal with the *What?* and not the *How?* in software projects. Hence, system-level requirements for QC do not substantially differ from classical requirements. However, new types of nonfunctional requirements specific to quantum software in combination with quantum hardware might become relevant.

Systems Design/Architecture (Hardware–Software–QC Co-Design) In this phase, the problem splitting between classical and quantum tasks takes place: the engineer decides which parts of the overall problem are solved by classical computations and which ones by quantum solutions. This requires architectural guidelines and patterns, as well as interface descriptions for interactions between classical and quantum data.

Programming Languages and Implementation (Q-Circuit) In this phase, algorithms need to be realized for the classical as well as for the quantum parts of a given problem. For the classical part, programming languages and compilation is well known. However, for implementing quantum algorithms, we currently rely on gate-level languages (even in case of seemingly higher-level quantum programming languages like Q#). While gate-level languages are essentially the quantum equivalent to classical assembly languages, for more efficient implementation, we need appropriate high-level quantum programming abstractions. Furthermore, we need programming guidelines and idioms, as well as design patterns for quantum programming languages. [107] Design by contract for quantum software.

Compilation and Deployment (Q-Circuit) Today, each quantum hardware comes with its own hardware specifics, e.g., gates that can be implemented easily or at all and to which qubits these gates can apply. This requires machine-specific compilation and transpilation techniques. OpenQASM is only becoming a de facto standard for hardware-level quantum programming. In order to allow for more efficient development and execution of quantum programs, we need a common intermediate language, e.g., OpenQASM, and generic compilation techniques. This includes instruction set selection and back-end optimization that can be easily adapted and configured for specific hardware. Here, ideas for classical compiler-compilers may become useful again to automatically generate hardware-specific compilers.

This aspect naturally includes devising new methods to (statically) check desirable properties and guarantees of quantum programs at compile time; the first steps in this direction have already been taken [112, 47, 106, 72].

Testing and Verification In the spirit of the V-model and similar development models, the approaches in this phase complement the approaches of the respective development phases. Recent techniques for testing and verification of (partly) probabilistic hybrid software systems may provide a conceptual foundation for ensuring that the observed output behavior of quantum components conforms to a given specification [41, 62, 109, 35, 2]. Corresponding black-box techniques are applicable at the functional unit level as well as the system integration level of the hybrid system, by abstracting from any internal details of quantum components (application scenarios 1 and 3). In contrast, in the case of a white-box setting (application scenario 2), it is not obvious how to adopt established techniques for software testing (e.g., interactive debugging [62]) and verification of quantum computations. The first step in this direction may be to find sound abstractions that properly reflect quantum-specific phenomena like superposition and entanglement of computational states and the destructive nature of quantum measurements. As always, testing and verification aim at improving software quality and minimizing the number of bugs; the first steps in the direction of understanding the quantum-specific aspects of these goals have been taken [113].

3.2 Perils

Quantum computing will benefit from established software engineering techniques. The synthesis of both fields will likely put a few new topics on the joint research agenda. However, there is also good reason to predict that quantum software engineering will (a) likely not radically change most established means of software engineering and (b) not benefit from inapt, straightforward adaptations of existing insights. In particular, we argue that this concerns the use of modeling languages and adaptations of development processes.

Albeit special-purpose quantum languages are available, most development activities in the NISQ either comprise using quantum functionalities on the API level or constructing gate sequences that are applied on qubits. Dispatching and orchestration aspects are embedded into a classical host scripting language, typically Python [71, 91]. The translation between different APIs is currently near-trivial [83]. Special-purpose quantum programming languages (or extensions to classical languages) promise to lift the specification or verification of quantum *algorithms* to more appropriate levels of abstraction that require less manual handling of details. We are not aware of an argument as to why abstraction levels that transcend algorithmic implementation details, and thus avoid quantum specifics, would necessarily need to be crafted differently than in the classical case. Of course, it is possible to use mechanisms like UML that were intended to model software designs for describing low-level details of qubits, quantum registers, and gates. Yet it would also be possible to model classical bits, registers and electronic gates using UML in the same way; since we are not aware of any beneficial application of such a technique to the best of our knowledge, this underlines the importance of not mixing modeling techniques targeted at high levels of abstraction with low-level details. While the *design* of algorithms for quantum systems is entirely different from classical algorithms (and systematic methods range among the most challenging unsolved problems in the field), implementation details in general almost never concern modeling at a higher level [10], and should therefore continue not to do so in quantum software development.

Again to the best of our knowledge, using entirely nonstandard development processes in specific domains is not commonly reported in the literature. Likewise, we are unaware of specially crafted software development processes—unlike architectures—that are beneficial when components like GPU accelerators or target domains like cloud deployments are considered. As we have argued in the use case discussion above, GPUs can be seen as computational accelerators (in local appliances) or cloud resources (in distributed systems), which by analogy suggests that any such specially crafted processes will not lead to pronounced advantages. Additionally, software engineering research often finds little to no difference [59] when the implications of various forms of (social) process interactions between developers are studied for software in different domains. This insight further strengthens the hypothesis that quantum software development can be based on existing processes, and inherit the advantages and disadvantages of each approach.

Consequently, we find it unlikely that quantum software engineering in any of the scenarios described in this chapter will require entirely new development processes, or nontrivial modifications of existing approaches. Since no substantial body of quantum software exists yet, mining quantitative empirical evidence toward one side or another will likely not be conclusive at this stage. While it cannot be ruled out that, for instance, UML will be an appropriate tool to design algorithms at gate and qubit level, or that entirely new development processes will need to be devised to implement quantum software, we call for caution before making overly ambitious statements without conclusive evidence, which could either be derived from sound ab initio considerations or empirically observed from mounting industrial and academic experience with creating concrete quantum software.

4 Summary and Outlook

Quantum computing is still in a very early stage with major challenges ahead. Many of these challenges have to be addressed by advancing quantum computing at the hardware level. Nevertheless, quantum computing will not only be pushed by innovations in physics, leading to advancements in quantum hardware, but progress can also be expected by a pull effect caused by innovative future applications. Or, according to the aforementioned quote by Deshpande [28], realizing practical applications is indeed in the domain of SE.

Nevertheless, quantum software development will not cause a revolution in SE, neither today nor in the foreseeable future. The overall aim of many SE principles (e.g., separation of concerns, encapsulation, and information hiding, just to name a few) is exactly to be agnostic to diverse (existing and future) computational platforms. Hence, we should be more interested in those characteristics of quantum computations which have been exotic corner cases in SE until now but will soon become omnipresent in quantum software development.

Moreover, quantum software development today mostly happens at source code level and reaching downwards to assembly level. Quantum programming today mostly means to custom-tailor a quantum solution to a very specific instruction set of a specifically developed special-purpose quantum computer. The tendency in mainstream SE 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 SE research like compiler construction and instruction set architecture design will become highly relevant again.

Acknowledgments This work is supported by the German Federal Ministry of Education and Research within the funding program Quantum technologies—from basic research to market, contract numbers 13N16303 (IS) and 13N15647/13NI6092 (WM). WM also acknowledges support from the High-Tech Agenda of the Free State of Bavaria. The work is further partly supported by the Austrian ministries BMK, BMAW, and the State of Upper Austria in the frame of the FFG COMET competence center INTEGRATE at SCCH as well as by BMBWF via the project QuantumReady (RR). We also acknowledge the use of IBM Quantum services and the Fraunhofer quantum computer in Ehningen for this work (MJ).

References

1. Aaronson, S.: Quantum Computing Since Democritus. Cambridge University Press, USA (2013). ISBN:0521199565
2. Abreu, R., et al.: Metamorphic Testing of Oracle Quantum Programs. In: 2022 IEEE/ACM 3rd International Workshop on Quantum Software Engineering (Q-SE), pp. 16–23 (2022). https://doi.org/10.1145/3528230.3529189
3. Akshay, V., et al.: Reachability deficits in quantum approximate optimization. Phys. Rev. Lett. **124**(9), 090504 (Mar. 2020). https://doi.org/10.1103/PhysRevLett.124.090504. https://link.aps.org/doi/10.1103/PhysRevLett.124.090504
4. Alam, M., Ash-Saki, A., Ghosh, S.: Design-Space Exploration of Quantum Approximate Optimization Algorithm under Noise. In: 2020 IEEE Custom Integrated Circuits Conference (CICC), pp. 1–4 (2020). https://doi.org/10.1109/CICC48029.2020.9075903
5. Algaba, M.G., et al.: Co-design quantum simulation of nanoscale NMR. Phys. Rev. Res. **4**(4), 043089 (Nov. 2022). https://doi.org/10.1103/PhysRevResearch.4.043089. https://link.aps.org/doi/10.1103/PhysRevResearch.4.043089
6. Ali, S., Yue, T., Abreu, R.: When software engineering meets quantum computing. Commun. ACM **65**(4), 84–88 (Mar. 2022). ISSN:0001-0782. https://doi.org/10.1145/3512340
7. Altman, E., et al.: Quantum simulators: Architectures and opportunities. PRX Quantum **2**(1), 017003 (2021)
8. Antonino, P.O., et al.: Enabling Continuous Software Engineering for Embedded Systems Architectures with Virtual Prototypes. In: Cuesta, C.E., Garlan, D., Pérez, J. (eds.) Software Architecture, pp. 115–130. Springer International Publishing, Cham (2018). ISBN:978-3-030-00761-4
9. Awasthi, A., et al.: Quantum Computing Techniques for Multi-Knapsack Problems (2023). https://doi.org/10.48550/ARXIV.2301.05750. https://arxiv.org/abs/2301.05750
10. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. SEI Series in Software Engineering. Addison-Wesley (2003). ISBN:9780321154958
11. Bayerstadler, A., et al.: Industry quantum computing applications. EPJ Quantum Technol. **8**(1), (Nov. 2021). https://doi.org/10.1140/epjqt/s40507-021-00114-x. https://epjquantumtechnology.springeropen.com/track/pdf/10.1140/epjqt/s40507-021-00114-x.pdf
12. Becker, C.K.-U., Gheorghe-Pop, I.-D., Tcholtchev, N.: A Testing Pipeline for Quantum Computing Applications. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
13. Bharti, K., et al.: Noisy intermediate-scale quantum algorithms. Rev. Mod. Phys. **94**(1), 015004 (Feb. 2022). https://doi.org/10.1103/RevModPhys.94.015004. https://link.aps.org/doi/10.1103/RevModPhys.94.015004
14. Bichsel, B., et al.: 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, PLDI 2020, pp. 286–300. Association for Computing Machinery, London, UK (2020). ISBN:9781450376136. https://doi.org/10.1145/3385412.3386007
15. Bittner, T., Groppe, S.: Avoiding Blocking by Scheduling Transactions Using Quantum Annealing. In: Proceedings of the 24th Symposium on International Database Engineering & Applications, IDEAS '20. Association for Computing Machinery, Seoul, Republic of Korea (2020). ISBN:9781450375030. https://doi.org/10.1145/3410566.3410593
16. Blatt, R., Roos, C.F.: Quantum simulations with trapped ions. Nature Phys. **8**(4), 277–284 (2012)
17. Blekos, K., et al.: A Review on Quantum Approximate Optimization Algorithm and Its Variants (2023). arXiv:2306.09198 [quant-ph]
18. Buluta, I., Nori, F.: Quantum simulators. Science **326**(5949), 108–111 (2009). https://doi.org/10.1126/science.1177838. eprint: https://www.science.org/doi/pdf/10.1126/science.1177838. https://www.science.org/doi/abs/10.1126/science.1177838

19. Business Value II for: The Quantum Decade: A Playbook for Achieving Awareness, Readiness, and Advantage. IBM Institute for Business Value (2021). ISBN:9781737401100. https://books.google.de/books?id=MeN%5C_zgEACAAJ
20. Cirac, J.I., Zoller, P.: Goals and opportunities in quantum simulation. Nature Phys. **8**(4), 264–266 (2012). https://doi.org/10.1038/nphys2275
21. Cross, A., et al.: OpenQASM 3: A broader and deeper quantum assembly language. ACM Trans. Quantum Comput. **3**(3), (Sept. 2022). ISSN:2643-6809. https://doi.org/10.1145/3505636
22. Cubitt, T.S., Perez-Garcia, D., Wolf, M.M.: Undecidability of the spectral gap. Nature **528**(7581), 207–211 (2015). https://doi.org/10.1038/nature16059
23. D-Wave Systems Inc.: D-Wave Systems Leap Cloud Service (2023). https://cloud.dwavesys.com/leap/ visited 2023-03-03
24. De Andoin, M.G., et al.: Comparative Benchmark of a Quantum Algorithm for the Bin Packing Problem. In: 2022 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 930–937. IEEE (2022)
25. Deb, A., Dueck, G.W., Wille, R.: Exploring the potential benefits of alternative quantum computing architectures. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **40**(9), 1825–1835 (2020)
26. Delgado, A., et al.: Simulating key properties of lithium-ion batteries with a fault-tolerant quantum computer. Phys. Rev. A **106**(3), 032428 (Sept. 2022). https://doi.org/10.1103/PhysRevA.106.032428. https://link.aps.org/doi/10.1103/PhysRevA.106.032428
27. Deng, Y.-H., et al.: Solving graph problems using Gaussian Boson sampling. Phys. Rev. Lett. **130**(19), 190601 (May 2023). https://doi.org/10.1103/PhysRevLett.130.190601. https://link.aps.org/doi/10.1103/PhysRevLett.130.190601
28. Deshpande, A.: Assessing the quantum-computing landscape. Commun. ACM **65**(10), 57–65 (2022)
29. Dey, N., et al.: QDLC – The Quantum Development Life Cycle (2020). arXiv:2010.08053 [cs.ET]
30. Ding, Y., et al.: Systematic Crosstalk Mitigation for Superconducting Qubits via Frequency-Aware Compilation. In: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 201–214 (2020). https://doi.org/10.1109/MICRO50266.2020.00028
31. Elben, A., et al.: The randomized measurement toolbox. Nature Rev. Phys. **5**(1), 9–24 (2023). https://doi.org/10.1038/s42254-022-00535-2
32. Evans, A., et al.: MCBeth: A Measurement Based Quantum Programming Language (2022). arXiv:2204.10784 [cs.PL]
33. Farhi, E., Goldstone, J., Gutmann, S.: A Quantum Approximate Optimization Algorithm (2014). https://doi.org/10.48550/ARXIV.1411.4028. https://arxiv.org/abs/1411.4028
34. Fedorov, D.A., et al.: VQE method: a short survey and recent developments. Mater. Theory **6**(1), 2 (2022). https://doi.org/10.1186/s41313-021-00032-6
35. Feng, Y., Duan, R., Ying, M.: Bisimulation for quantum processes. ACM Trans. Program. Lang. Syst. **34**(4), (2012). https://doi.org/10.1145/2400676.2400680
36. Fortnow, L.: One Complexity Theorist's View of Quantum Computing. In: Electronic Notes in Theoretical Computer Science. 31 CATS 2000 Computing: the Australasian Theory Symposium, pp. 58–72 (2000). ISSN:1571-0661. https://doi.org/10.1016/S1571-0661(05)80330-5. https://www.sciencedirect.com/science/article/pii/S1571066105803305
37. Furutanpey, A., et al.: Architectural Vision for Quantum Computing in the Edge-Cloud Continuum. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
38. Garcìa de la Barrera, A., et al.: Quantum software testing: State of the art. J. Software Evol. Process **35**(4), e2419 (2023). https://doi.org/10.1002/smr.2419. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2419
39. Gemeinhardt, F.G., Wille, R., Wimmer, M.: Quantum k-community detection: algorithm proposals and cross-architectural evaluation. Quantum Inf. Process. **20**(9), 302 (2021)

40. Georgescu, I.M., Ashhab, S., Nori, F.: Quantum simulation. Rev. Modern Phys. **86**(1), 153 (2014)
41. Gerhold, M., Stoelinga, M.: Model-based testing of probabilistic systems. Formal Aspects Comput. **30**(1), 77–106 (Jan. 2018). ISSN:0934-5043. https://doi.org/10.1007/s00165-017-0440-4
42. Girbal, S., et al.: On the convergence of mainstream and mission-critical markets. In: 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–10 (May 2013). https://doi.org/10.1145/2463209.2488962
43. Greiwe, F., Krüger, T., Mauerer, W.: Effects of Imperfections on Quantum Algorithms: A Software Engineering Perspective. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
44. Hangleiter, D., Eisert, J.: Computational advantage of quantum random sampling. Rev. Modern Phys. **95**(3), (July 2023). https://doi.org/10.1103/revmodphys.95.035001
45. Heckemann, K., et al.: Safe Automotive Software. In: König, A., et al. (eds.) Knowledge-Based and Intelligent Information and Engineering Systems, pp. 167–176. Springer, Berlin, Heidelberg (2011). ISBN:978-3-642-23866-6
46. Herrmann, N., et al.: Quantum Utility—Definition and Assessment of a Practical Quantum Advantage. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
47. Klamroth, J., et al.: QIn: Enabling Formal Methods to Deal with Quantum Circuits. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
48. Kole, A., et al.: Improved mapping of quantum circuits to IBM QX architectures. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **39**(10), 2375–2383 (2019)
49. Kreppel, F., et al.: Quantum Circuit Compiler for a Shuttling-Based Trapped-Ion Quantum Computer (2022). https://doi.org/10.48550/ARXIV.2207.01964. https://arxiv.org/abs/2207.01964
50. Krüger, T., Mauerer, W.: Quantum Annealing-Based Software Components: An Experimental Case Study with SAT Solving (2020). Q-SE@ICSE. https://arxiv.org/abs/2005.05465
51. Lamata, L., et al.: Digital-analog quantum simulations with superconducting circuits. Adv. Phys. X **3**(1), 1457981 (2018). https://doi.org/10.1080/23746149.2018.1457981
52. Li, G., et al.: On the Co-Design of Quantum Software and Hardware. In: Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication NANOCOM '21 Association for Computing Machinery, Virtual Event, Italy (2021). ISBN:9781450387101. https://doi.org/10.1145/3477206.3477464
53. Liu, J., et al.: Noise can be helpful for variational quantum algorithms (Oct. 2022). arXiv: 2210.06723 [quant-ph]
54. Lloyd, S.: Universal quantum simulators. Science **273**(5278), 1073–1078 (1996). https://doi.org/10.1126/science.273.5278.1073. eprint: https://www.science.org/doi/pdf/10.1126/science.273.5278.1073. https://www.science.org/doi/abs/10.1126/science.273.5278.1073
55. Lobe, E., Stollenwerk, T.: QUARK (Feb. 2022). https://quantum-computing-software.gitlab.io/quark/
56. Lubinski, T., et al.: Advancing hybrid quantum-classical computation with real-time execution. Front. Phys. **10**, (2022). ISSN:2296-424X. https://doi.org/10.3389/fphy.2022.940293. https://www.frontiersin.org/articles/10.3389/fphy.2022.940293
57. Marwedel, P.: Embedded System Design - Embedded Systems Foundations of Cyber-Physical Systems, Second Edition. Embedded Systems Springer (2011). ISBN: 978-94-007-0256-1. https://doi.org/10.1007/978-94-007-0257-8
58. Mauerer, W.: Semantics and simulation of communication in quantum programming (2005). https://doi.org/10.48550/ARXIV.QUANT-PH/0511145. https://arxiv.org/abs/quant-ph/0511145

59. Mauerer, W., Joblin, M., et al.: In search of socio-technical congruence: A large-scale longitudinal study. IEEE Trans. Software Eng. (01), 1–1 (May 2021). ISSN: 1939-3520. https://doi.org/10.1109/TSE.2021.3082074. https://www.computer.org/csdl/journal/ts/5555/01/09436025/1tJsglfkGru
60. Mauerer, W., Scherzinger, S.: 1-2-3 Reproducibility for Quantum Software Experiments. Q-SANER@IEEE International Conference on Software Analysis, Evolution and Reengineering (2022)
61. Medvidović, M., Carleo, G.: Classical variational simulation of the quantum approximate optimization algorithm. npj Quantum Inf. **7**(1), 101 (2021)
62. Miranskyy, A., Zhang, L.: On Testing Quantum Programs. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), pp. 57–60 (2019). https://doi.org/10.1109/ICSE-NIER.2019.00023
63. Morgado, M., Whitlock, S.: Quantum simulation and computing with Rydberg-interacting qubits. AVS Quantum Sci. **3**(2), 023501 (June 2021). https://doi.org/10.1116/5.0036562
64. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
65. Pashayan, H., Wallman, J.J., Bartlett, S.D.: Estimating outcome probabilities of quantum circuits using quasiprobabilities. Phys. Rev. Lett. **115**(7), 070501(Aug. 2015). https://doi.org/10.1103/PhysRevLett.115.070501. https://link.aps.org/doi/10.1103/PhysRevLett.115.070501
66. Paudel, H.P., et al.: Quantum computing and simulations for energy applications: review and perspective. ACS Eng. Au **2**(3), 151–196 (2022). https://doi.org/10.1021/acsengineeringau.1c00033
67. Peham, T., Burgholzer, L., Wille, R.: Equivalence checking paradigms in quantum circuit design: a case study. In: Oshana, R. (ed.) DAC '22: 59th ACM/IEEE Design Automation Conference, San Francisco, California, USA, July 10–14, 2022, pp. 517–522. ACM (2022). https://doi.org/10.1145/3489517.3530480
68. Perdomo-Ortiz, A., et al.: A quantum annealing approach for fault detection and diagnosis of graph-based systems. Eur. Phys. J. Special Top. **224**, 131–148 (2015)
69. Pérez-Delgado, C.A., Perez-Gonzalez, H.G.: Towards a Quantum Software Modeling Language. In: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops ICSEW'20, pp. 442–444. Association for Computing Machinery, Seoul, Republic of Korea (2020). ISBN:9781450379632. https://doi.org/10.1145/3387940.3392183
70. Poggel, B., et al.: Recommending Solution Paths for Solving Optimization Problems with Quantum Computing. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
71. Qiskit Contributors.: Qiskit: An Open-Source Framework for Quantum Computing (2023). https://doi.org/10.5281/zenodo.2573505
72. Quetschlich, N., Burgholzer, L., Wille, R.: Predicting good quantum circuit compilation options. In: 2023 IEEE International Conference on Quantum Software (QSW), pp. 43–53. IEEE (2023)
73. Ramsauer, R., et al.: Static Hardware Partitioning on RISC-V - Shortcomings, Limitations, and Prospects. In: 8th IEEE World Forum on Internet of Things (IEEE WFIoT2022) (July 2022). https://doi.org/10.48550/arXiv.2208.02703. https://arxiv.org/abs/2208.02703
74. Reich, J., Schneider, D., et al.: Engineering of Runtime Safety Monitors for Cyber-Physical Systems with Digital Dependability Identities. In: Casimiro, A., et al. (eds.) Computer Safety, Reliability, and Security, pp. 3–17. Springer International Publishing, Cham (2020). ISBN:978-3-030-54549-9
75. Reich, J., Wellstein, M., et al.: Towards a Software Component to Perform Situation-Aware Dynamic Risk Assessment for Autonomous Vehicles. In: Adler, R., et al. (eds.) Dependable Computing - EDCC 2021 Workshops, pp. 3–11. Springer International Publishing, Cham (2021). ISBN:978-3-030-86507-8

76. Resch, S., Karpuzcu, U.R.: Benchmarking quantum computers and the impact of quantum noise. ACM Comput. Surv. **54**(7), (July 2021). ISSN:0360-0300. https://doi.org/10.1145/3464420
77. Rice, J.E., et al.: Quantum computation of dominant products in lithium–sulfur batteries. J. Chem. Phys. **154**(13), 134115 (Apr. 2021). ISSN:0021-9606. https://doi.org/10.1063/5.0044068. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0044068/15588046/134115_1_online.pdf
78. Roffe, J.: Quantum error correction: an introductory guide. Contemp. Phys. **60**(3), 226–245 (2019). https://doi.org/10.1080/00107514.2019.1667078
79. Safi, H., Wintersperger, K., Mauerer, W.: Influence of HW-SW-Co-Design on Quantum Computing Scalability. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
80. Saurabh, N., Jha, S., Luckow, A.: A Conceptual Architecture for a Middleware for Hybrid Quantum-HPC Application Workflows. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
81. Sax, I., et al.: Approximate Approximation on a Quantum Annealer. In: Proceedings of the 17th ACM International Conference on Computing Frontiers, pp. 108–117 (2020). https://arxiv.org/pdf/2004.09267
82. Schmale, T., et al.: Backend compiler phases for trapped-ion quantum computers. In: 2022 IEEE International Conference on Quantum Software (QSW), pp. 32–37. IEEE Computer Society, Los Alamitos, CA, USA (July 2022). https://doi.org/10.1109/QSW55613.2022.00020. https://doi.ieeecomputersociety.org/10.1109/QSW55613.2022.00020
83. Schönberger, M., Franz, M., et al.: Peel — Pile? Cross-Framework Portability of Quantum Software. In: 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C), pp. 164–169 (2022). https://doi.org/10.1109/ICSA-C54293.2022.00039
84. Schönberger, M., Scherzinger, S., Mauerer, W.: Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. In: Proceedings of ACM SIGMOD/PODS International Conference on Management of Data (2023)
85. Schönberger, M., Trummer, I., Mauerer, W.: Quantum Optimisation of General Join Trees. In: Proceedings of the International Workshop on Quantum Data Science and Management, QDSM '23 (Aug. 2023)
86. Schreiber, F.J., Eisert, J., Meyer, J.J.: Classical surrogates for quantum learning models (2022). arXiv: 2206.11740 [quant-ph]
87. Schuld, M., Sweke, R., Meyer, J.J.: Effect of data encoding on the expressive power of variational quantum-machine-learning models. Phys. Rev. A **103**(3), 032430 (Mar. 2021). https://doi.org/10.1103/PhysRevA.103.032430. https://link.aps.org/doi/10.1103/PhysRevA.103.032430
88. Schulz, M., et al.: Accelerating HPC with quantum computing: It is a software challenge too. Comput. Sci. Eng. **24**(04), 60–64 (July 2022). ISSN:1558-366X. https://doi.org/10.1109/MCSE.2022.3221845
89. Serrano, M.A., Perez-Castillo, R., Piattini, M., (eds.): Quantum Software Engineering. Springer (2022). ISBN:978-3-031-05323-8. https://doi.org/10.1007/978-3-031-05324-5
90. Sitdikov, I., et al.: Middleware for Quantum: An orchestration of hybrid quantum-classical systems. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
91. Sivarajah, S., et al.: t|ket⟩: a retargetable compiler for NISQ devices. Quantum Sci. Technol. **6**(1), 014003 (Nov. 2020). https://doi.org/10.1088/2058-9565/ab8e92
92. Sommerville, I.: Software Engineering, 9th edn. Addison-Wesley, Harlow, England (2010). ISBN:978-0-13-703515-1
93. Steiner, L., et al.: An LPDDR4 Safety Model for Automotive Applications. In: The International Symposium on Memory Systems MEMSYS 2021 Association for Computing Machinery, Washington DC, DC, USA (2022). ISBN:9781450385701. https://doi.org/10.1145/3488423.3519333

94. Stollenwerk, T., Lobe, E., Jung, M.: Flight gate assignment with a quantum annealer. In: International Workshop on Quantum Technology and Optimization Problems, pp. 99–110. Springer, Berlin (2019). https://elib.dlr.de/123777/. https://doi.org/10.1007/978-3-030-14082-3_9
95. Stollenwerk, T., O'Gorman, B., et al.: Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. IEEE Trans. Intell. Transp. Syst. **21**(1), 285–297 (2019)
96. Streif, M., Leib, M.: Training the quantum approximate optimization algorithm without access to a quantum processing unit. Quantum Sci. Technol. **5**(3), 034008 (May 2020). https://doi.org/10.1088/2058-9565/ab8c2b
97. Tilly, J., et al.: The Variational Quantum Eigensolver: A review of methods and best practices. Physics Reports, 986 The Variational Quantum Eigensolver: a review of methods and best practices, pp. 1–128 (2022). ISSN:0370-1573. https://doi.org/10.1016/j.physrep.2022.08.003. https://www.sciencedirect.com/science/article/pii/S0370157322003118
98. Trummer, I., Koch, C.: Multiple query optimization on the D-wave 2X adiabatic quantum computer. Proc. VLDB Endow. **9**(9), 648–659 (May 2016). ISSN:2150-8097. https://doi.org/10.14778/2947618.2947621
99. Ufrecht, C., et al.: Cutting multi-control quantum gates with ZX calculus (2023). https://doi.org/10.48550/ARXIV.2302.00387. https://arxiv.org/abs/2302.00387
100. Wang, S., et al.: Noise-induced barren plateaus in variational quantum algorithms. Nature Commun. **12**(1), 6961 (2021). https://doi.org/10.1038/s41467-021-27045-6
101. Weder, B., et al.: Quantum software development lifecycle. Quantum Software Engineering, pp. 61–83. Springer (2022)
102. Weimer, H., et al.: Digital quantum simulation with Rydberg atoms. Quantum Inf. Process. **10**(6), 885 (2011). https://doi.org/10.1007/s11128-011-0303-5
103. Wille, R., Hillmich, S., Burgholzer, L.: Tools for quantum computing based on decision diagrams. ACM Trans. Quantum Comput. **3**(3), 1–17 (2022)
104. Wintersperger, K., Dommert, F., et al.: Neutral Atom Quantum Computing Hardware: Performance and End-User Perspective (2023)
105. Wintersperger, K., Safi, H., Mauerer, W.: QPU-System Co-Design for Quantum HPC Accelerators. In: Proceedings of the 35th GI/ITG International Conference on the Architecture of Computing Systems (Aug. 2022). Gesellschaft für Informatik
106. Xia, S., Zhao, J.: Static Entanglement Analysis of Quantum Programs (2023). arXiv: 2304.05049 [cs.SE]
107. Yamaguchi, M., Yoshioka, N.: Design by Contract Framework for Quantum Software (2023). arXiv: 2303.17750 [cs.CL]
108. Yarkoni, S., et al.: Multi-car paint shop optimization with quantum annealing. In: 2021 IEEE International Conference on Quantum Computing and Engineering (QCE), pp. 35–41. IEEE (2021)
109. Ying, M.: Toward automatic verification of quantum programs. Formal Aspects Comput. **31**(1), 3–25 (2019). https://doi.org/10.1007/s00165-018-0465-3
110. Yue, T., et al.: Challenges and Opportunities in Quantum Software Architecture. In: Weber, I., Kazman, R., Pellicione, P. (eds.) Software Architecture Research Roadmaps from the Community. Springer (2023)
111. Zhao, J.: Quantum Software Engineering: Landscapes and Horizons. CoRR. abs/2007.07047 (2020). arXiv: 2007.07047. https://arxiv.org/abs/2007.07047
112. Zhao, P., Wu, X., Li, Z., et al.: QChecker: Detecting Bugs in Quantum Programs via Static Analysis (2023). arXiv: 2304.04387 [cs.SE]
113. Zhao, P., Wu, X., Luo, J., et al.: An Empirical Study of Bugs in Quantum Machine Learning Frameworks. In: Proceedings of the IEEE International Conference on Quantum Software. IEEE (2023)
114. Zhou, L., et al.: Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices. Phys. Rev. X **10**(2), 021067 (June 2020). https://doi.org/10.1103/PhysRevX.10.021067. https://link.aps.org/doi/10.1103/PhysRevX.10.021067