Quantum Software

Executive Summary

This NESSI[1] position paper reviews the current state of software technologies of quantum computing and highlights the strategic importance of addressing the research challenges identified.

Advances in the development of quantum computing hardware must be accompanied by advances on the software side in order to make quantum computing feasible. This was the central statement of the first NESSI paper on quantum computing published in 2022[2]. Since then, there has been good progress in the field of quantum software. Many of the major players have published their software (typically as SDKs) and an increasing number of online resources and self-paced learning courses are available to support better understand of the fundamental aspects of quantum computing. However, significant challenges still lie ahead to advance quantum software to the level of efficiency and sophistication that we see (and take for granted) in the classical computing world. In order to address these challenges and to ensure European technology leadership in quantum computing, NESSI proposes the following key actions.

- Identify the near-term business opportunities offered by quantum computing, and promote industry involvement to help seize these opportunities.
- Increase the number of computer scientists and software experts in quantum computing research and leverage their extensive knowledge in software design automation.
- Promote interdisciplinary collaboration between hardware experts and tool developers to improve the uptake of quantum computing across a diverse range of domains.
- Establish a cohesive quantum software community to increase efficiency in quantum software research and development and nurture an innovative quantum ecosystem.
- Conduct research addressing the software challenges of today's noisy quantum devices and preparing quantum software engineering (QSE) for the future era of large-scale fault-tolerant quantum computers (FTQCs).
- Conduct basic research to find quantum algorithms with better end-to-end complexities than their classical counterparts.

These actions should address the following key challenges.

- Adapting classical software engineering concepts and techniques. How can e.g. modularity, reusability and agile development be evolved to address the unique requirements of Quantum Software Engineering (QSE)?
- Developing effective and appropriate QSE tools. What tools can be created which cope with the inherent complexity of quantum mechanics and increase the efficiency in designing and developing quantum software solutions, including high-level programming languages which abstract much of the complexity of quantum mechanics and the different quantum devices?
- Integrating hybrid software systems. What methods and frameworks can enable seamless integration of quantum and classical software components into hybrid systems?
- Advancing quantum algorithms. What are the key steps in quantum algorithm development, including quantum-inspired and hybrid algorithms that combine classical and quantum elements?
- Enhancing classical software engineering through quantum computing. How can quantum computing be leveraged to advance classical software engineering practices, such as analysing, testing, and verifying large-scale and complex software systems?

1 NESSI (Networked European Software and Services Initiative), the European association promoting research, development and innovation in the field of software, data and digital services; https://nessi.eu/

2 "Software and Quantum Computing", NESSI (2022); https://nessi1.wpengine.com/wp-content/uploads/2022/05/NESSI-Quantum-Computing-issue-1.pdf

Introduction

Today's quantum computing is still in an early stage of development, with quantum computers being described as noisy intermediate-scale quantum (NISQ) devices. These devices have a relatively small number of qubits, and their performance is limited because of qubit decoherence and high error rates in quantum gate operations and measurements. There is the general expectation that advanced algorithms, such as Shor's algorithm for finding the prime factors of an integer, will require large-scale fault-tolerant quantum computers. FTQCs are quantum systems capable of performing reliable computations even in the presence of hardware imperfections and environmental noise, by employing error correction techniques to maintain the integrity of quantum information. It is generally expected that FTQCs will become available in the 2030's at the earliest. From that we might conclude that using quantum computers for solving business relevant problems is still far away in the future and proper quantum software engineering is not yet urgently needed. However, this conclusion is incorrect.

There are too few quantum software experts, and we lack a cohesive quantum software community for knowledge sharing as well as interdisciplinary collaboration between hardware experts and computer scientists. Despite these constraints, we can already observe early practical benefits from using today's quantum computing resources, including but not limited to the following.

- Quantum-as-a-Service offers access to quantum compute resources, enabling research and industry to experiment and develop novel quantum solutions.
- Hybrid quantum computing refers to utilising and combining both quantum and classical compute resources for building applications. For example, it covers algorithms that can cope well with the shortcomings of NISQ devices by using classical techniques for fine-tuning quantum circuits.
- Quantum-inspired computing provides a bridge to the benefits of quantum computing principles today while preparing for future FTQCs.

However, for future significant advancements in quantum software engineering, further development is still needed to unlock the full potential of both NISQ and FTQC devices[3].

Where is Quantum Software Today?

While quantum computers long remained a theoretical concept and a field of basic research, recently there has been tremendous progress in developing real quantum computers of continuously increasing size and quality. Despite these hardware developments, the software for quantum computing is still in its infancy compared to the sophisticated design tools and flows we take for granted in the classical realm.

There are many design steps and automation requirements that need to be addressed to make practical applications of quantum computing a reality. These include, but are not limited to, encoding the application into a quantum algorithm, simulating that algorithm, compiling it considering error correction, verifying its correctness, and mapping the problem to the chosen hardware.

Driven by recent advances, the first software solutions tackling these challenges are now available, notable examples being SDKs from major players such as IBM Qiskit, Amazon AWS Braket, Microsoft Azure Quantum and Google Cirq. However, the current state of quantum software still leaves much to be desired.

[3] Murillo, J. M., Garcia-Alonso, J., Moguel, E., Barzen, J., Leymann, F., Ali, S., ... & Wimmer, M. (2024) 'Challenges of quantum software engineering for the next decade: the road ahead', arXiv preprint arXiv:2404.06825.

More Computer Science Expertise Needed

Currently, most quantum software is not created by computer scientists or software experts, and therefore often lacks the power and efficiency of design automation solutions that are standard in classical computing. A survey at the Munich Quantum Software Forum[4] in 2023 revealed that only 39% of those working on quantum software identified themselves as having a background in computer science; most quantum software is still being developed by physicists (Figure 1). Whilst we acknowledge that the success of quantum computing requires an interdisciplinary effort across all fields, we believe that software development should be primarily driven by computer scientists and software experts who have extensive experience in design automation over many decades.

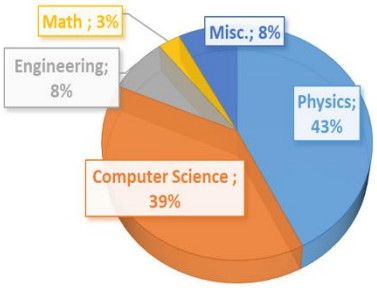

Figure 1 Survey results: educational background of experts involved in quantum software

Better Connection Between Tool Developers and Hardware Experts

Achieving success in quantum computing involves more than merely providing hardware platforms and design automation software. As these systems transition from laboratories to production environments, comprehensive ecosystems are essential, broadening the target audience from individual physics experts to a diverse range of domain users, many of whom may not be deeply familiar with quantum mechanics. With dozens or even hundreds of researchers from various disciplines involved in these fields, significant mediation between interdisciplinary stakeholders is required. Tool developers and hardware experts need to develop common terminology, definitions and frameworks that are universally understood, facilitating interdisciplinary collaboration[5]. This mirrors the role of classical design automation in bridging the gap between fields such as electrical engineering and computer science. It demands more than "interdisciplinarity on paper", calling for genuine interdisciplinary collaboration, demonstrated through tangible and impactful outputs.

Community Building

Currently, there is no cohesive quantum software community, with key players scattered across various silos. Additionally, the community consists of individuals with varying levels of maturity and expertise; some have been working in the domain for years, while others are just starting out. To build a high-quality professional community, it is crucial to integrate all players, ensuring that experienced members contribute their knowledge, while newcomers are welcomed and properly guided. This will allow us to identify the most promising approaches to building effective quantum software and solutions, and will enhance the competitiveness and innovation of a European quantum software community.

Quantum Software Engineering and Tool Support

Quantum software engineering is essential to exploit the full potential of quantum computing, transforming it from a purely theoretical concept into a practical technology capable of solving real-world problems. Without QSE, the adoption of quantum computing will be hampered by steep learning curves as well as high error rates and inefficiencies in software development, limiting its accessibility and applicability. To overcome these obstacles, QSE must provide software development methodologies, frameworks and tools that cope with the specificities and complexities of quantum computing, and guide the design, development, testing and maintenance of quantum applications.

[4] https://www.cda.cit.tum.de/research/quantum/mqsf/
[5] Schmid, L., Locher, D., Rispler, M., Blatt, S., Zeiher, J., Müller, M. & Wille, R. (2024) 'Computational Capabilities and Compiler Development for Neutral Atom Quantum processors—Connecting Tool Developers and Hardware Experts', Quantum Science and Technology (QST).

While the unique nature of quantum computing makes quantum software engineering fundamentally different from classical software engineering and thus requires quantum software specific approaches, we should avoid re-inventing the wheel and utilise experiences, solutions, and best practices from classical software engineering whenever possible. For example, quantum software design should support modularity and reusability, and quantum software development processes should be iterative, incremental and agile, as we know it from classical software engineering. However, the underlying models of classical and quantum computing are radically different, so classical software engineering techniques cannot be applied to QSE directly without significant changes.

Techniques designed for classical computing are based on an underlying model in which a sequence of instructions manipulates data represented in a circuit, and the final state of that data constitutes the output of the program. Quantum computing does not work with a sequence of instructions but with a system that can have multiple possible states at the same time. The computational process ends when a subset of those states collapses into a desired state, defining the final state of the entire system. Research is needed to revise classical software engineering methodologies to meet the specific needs of the quantum software development process and of the way quantum software design is produced from a requirement specification.

It will be the task of research and open-source projects to develop appropriate tools to support QSE and to bridge the gap between complex quantum theories and their application. Quantum algorithms are often represented as intricate circuits which must be optimised for execution on today's noisy, resource-constrained quantum hardware. Advanced tools are needed to automate circuit design, optimise gate sequences, and manage qubit allocation, while considering hardware constraints such as decoherence and connectivity. In addition, as quantum programs grow in size and complexity, tool support for debugging, simulation, verification and error correction becomes indispensable to ensure correctness and reliability. For example, tools that make use of optimised data structures and advanced methods, such as decision diagrams or tensor networks, can be crucial to efficiently handle quantum states and operations.

Tools such as the Munich Quantum Toolkit[6] can significantly reduce the complexity of quantum and hybrid software design and development, minimise human errors, and enable developers to create robust applications more quickly and efficiently. They must also meet the needs of developers, physicists, and end-users, ease their onboarding into the quantum domain, and support a broad uptake of quantum computing.

Quantum-as-a-Service

Today's information systems are complex structures formed by a combination of simple components, globally distributed and connected to each other by means of communication infrastructures and protocols. These components, often encapsulated as services, have specific functions, such as performing calculations, storing information, or processing data which is then sent to other services.

Service-Oriented Computing (SOC) and Cloud Computing have transformed software system architecture by introducing loosely coupled independent services communicating seamlessly over networks. SOC pioneered the concept of building systems from modular services, while cloud computing provided the necessary scalable infrastructure and on-demand computing power to deploy, manage, and expand these services globally. Together, these paradigms have been revolutionary, offering dynamic resource allocation and reduced operational costs through pay-per-use models together with enhanced reliability and availability. This has enabled systems with high-quality attributes such as interoperability, platform independence, minimised coupling, and maximised reusability of components. The modular structure of service-based architectures also improves maintainability by allowing individual components to be updated or replaced

[6] https://www.cda.cit.tum.de/research/quantum/mqt/

without disrupting the entire system. Furthermore, the inherent scalability of these architectures supports growth and adaptability as requirements change.

Quantum-as-a-Service (QaaS) is an emerging paradigm which enables developers to access quantum computing resources through cloud-based platforms. QaaS democratises access to quantum technologies, allowing users to experiment with, develop, and deploy quantum algorithms without necessarily needing direct access to expensive quantum hardware. This model leverages the strengths of SOC by providing a scalable, flexible and cost-effective framework for integrating quantum and classical computing resources[7]. As quantum hardware continues to evolve, QaaS has the potential to accelerate the adoption of quantum computing by lowering barriers to entry, enabling more diverse applications, and supporting hybrid quantum-classical workflows. QaaS is poised to play a pivotal role in driving the next phase of quantum innovation, bridging the gap between theoretical research and practical implementations.

Moreover, QaaS is a strategic model that facilitates the integration of quantum computing into existing classical IT infrastructures, providing organisations and developers with access to powerful quantum resources without the need for specialised hardware or quantum expertise. The high cost, technical complexity and maintenance requirements associated with quantum hardware make it infeasible for most enterprises to adopt quantum technologies directly. QaaS addresses these barriers by offering a cloud-based approach that provides quantum capabilities on-demand, enabling enterprises to experiment with, develop and deploy enhanced quantum solutions without investing in physical infrastructure.

Despite these benefits, quantum hardware and quantum software engineering methodologies have not yet reached the level of abstraction needed to treat quantum software as fully integrated services, as is the case in classical computing. Aspects such as interoperability, platform independence and capacity management are still far from being addressed for quantum software. Further research is needed on how to unify these two domains, so that the principles of SOC can be fully applied in the context of quantum computing[8].

Hybrid Quantum Computing

Hybrid quantum computing is a particularly promising way to link quantum computing to potential end-user applications. While the term is used to describe several different paradigms, we will discuss hybrid quantum computing in terms of variational quantum-classical applications and algorithm design, as well as approaches to building hybrid quantum-classical applications which utilise best-in-class quantum and high-performance classical resources.

Whilst state-of-the-art quantum processors remain modest in size with appreciable error rates, variational quantum algorithms (VQAs) have long stood out for their reasonable tolerance to noise and their potential applications with a limited number of qubits. VQAs are well-suited for solving problems across various scientific and industrial domains. These algorithms involve a hybrid quantum-classical approach in which a quantum circuit is parameterised, and these parameters are optimised using classical optimisation techniques. Commonly used examples of VQAs following this structure include the Variational Quantum Eigensolver (VQE)[9] or Quantum Approximate Optimisation Algorithm (QAOA)[10]. However, VQAs have

[7] Romero-Álvarez, J., Alvarado-Valiente, J., Moguel, E., Garcia-Alonso, J., & Murillo, J. M. (2024) 'Quantum Service-oriented Computing: A Proposal for Quantum Software as a Service', CRC Press.
[8] Moguel, E., Rojo, J., Valencia, D., Berrocal, J., Garcia-Alonso, J., & Murillo, J. M. (2022). Quantum service-oriented computing: current landscape and challenges. Software Quality Journal, 30(4), 983-1002.
[9] Peruzzo, A., McClean, J., Shadbolt, P., Yung, M-H, Zhou, X-Q, Love, P.J., Aspuru-Guzik,A. & O'Brien, J.L.. (2014) 'A variational eigenvalue solver on a photonic quantum processor', Nature communications, 5:4213.
[10] Farhi, E., Goldstone, J. & Gutmann, S. (2014) 'A quantum approximate optimization algorithm', arXiv preprint arXiv:1411.4028.

recently been criticised for known limitations navigating complicated optimization landscapes towards convergence.

A wider family of hybrid quantum-classical algorithms are starting to emerge which use classical subroutines to handle tasks beyond pure optimisation of parameters, such as data preparation or post-processing for experiments run on quantum computers, e.g. optimised circuit design. There is cutting-edge research into using classical state preparation and post-processing techniques, including quantum-inspired methods, which utilise best-in-class high performance computing resources to solve problems up to the limit of classical computational capacity[11]. AI methods including reinforcement learning tools are being deployed to explore the landscape of optimal efficient compilation or transpilation of problems onto state-of-the-art hardware, as well as in support of writing and coding quantum programmes. Furthermore, post-processing techniques, including error mitigation strategies, often demand significant classical compute capacity. All these examples, which are by no means exhaustive of the possibilities for classical computing to support the realisation of useful work on near-term quantum hardware, are key to so-called quantum utility experiments[12].

More generally, we can explore the design of hybrid quantum computing applications not just as a feature of near-term quantum computing application research but also as a key feature that will prevail in a fault-tolerant era. Quantum computing leverages the principles of quantum mechanics to form the basis of an entirely different approach to information processing. It is hoped that this fundamentally different approach to computation could help to solve problems that are considered classically intractable. However, quantum computing is not expected to be good at every type of classically difficult problem. Quantum computing is not expected to ever supersede the need for classical computing; rather, it is envisaged as an accelerator working together with best-in-class classical accelerators to tackle the most difficult problems. Quantum computers are often described as quantum processing units (QPUs), to highlight that they interact with classical processors in much the same was as graphics processing units (GPUs) do.

Technologies, software tools and frameworks anticipating this need are beginning to appear increasingly frequently across classical and quantum research fields. Many research groups and software organisations are providing important thought-leadership on how to most efficiently address computationally difficult problems, divide them into subroutines and delegate those subroutines to the most appropriate hardware, whether quantum or classical. Software engineering plays a core role, including in workload management, circuit cutting and knitting tools, intelligent orchestration, problem reconstruction, etc.

Key research challenges over the next few years include further understanding of where best to use classical and quantum resources as part of efficient quantum algorithm design. Hybrid algorithm design will depend on sophisticated software tooling to delegate many tasks such as state preparation and post-processing of quantum experiments, or in dedicated subroutines of an algorithm, such as a classical optimization step. Finally, an open-research problem is a usable QaaS implementation to assist in the management of these hybrid workflows and the wider question of quantum-classical workflow orchestration.

Quantum-Inspired Computing

Quantum-inspired computing refers to a set of computational techniques and algorithms that draw inspiration from the principles of quantum mechanics without requiring quantum hardware. These methods emulate concepts such as superposition and entanglement to enhance classical computing capabilities.

[11] Robledo-Moreno, J. et al (2024) 'Chemistry beyond exact solutions on a quantum-centric supercomputer', arXiv preprint arXiv:2405.05068.
[12] Kim, Y., Eddins, A., Anand, S. et al (2023) 'Evidence for the utility of quantum computing before fault tolerance', Nature 618, 500–505.

Sometimes quantum algorithms also admit a classical way of performing the same computation efficiently. Quantum-inspired approaches can be implemented on conventional computers, making them accessible and practical for a wide range of applications. In some cases, quantum-inspired algorithms replace a pure quantum approach, in other cases they serve as a tool for exploration, or even a way to assist in enabling useful near-term hybrid quantum-classical applications. As we await the full realisation of quantum computers, quantum-inspired methods provide a valuable bridge, allowing researchers and industries to benefit from quantum principles today while preparing for future quantum advancements. Moreover, quantum-inspired computing allows the exploration of quantum-like advantages without the need for expensive and complex quantum hardware and democratises access to advanced computational techniques.

Quantum-inspired computing, while promising, faces several challenges. Although these algorithms show potential, scalability for real-world applications can be a significant issue, especially maintaining performance as problem sizes grow. Additionally, there is limited quantum advantage: while quantum-inspired techniques can outperform classical algorithms, they may not fully capture the advantages of quantum computing itself. Understanding the boundaries and limitations of these approaches is crucial. Another challenge lies in the complexity of implementation of quantum-inspired algorithms, which often require a deep understanding of both quantum principles and classical computing techniques. This complexity can deter practitioners from adopting these methods. Moreover, the evolution of quantum computing means that the landscape of quantum-inspired techniques continues to change. Keeping up with advances and integrating new findings into existing frameworks is essential for anyone working in this area.

Developing quantum algorithms is essential because they can solve certain large problems considered intractable for classical computers, offering significant advantages in efficiency and performance[13]. Quantum algorithms not only push computational boundaries but also enhance our understanding of quantum-inspired methods. By advancing quantum algorithms, we can drive innovation and create a future in which classical and quantum computing work together effectively.

Quantum Software in the Future

Quantum software is starting to provide bridges between hardware providers and potential end users. It is enabling hybrid systems integrating quantum and classical components, using dedicated tooling, managing hybrid workflows via QaaS, and pushing the boundaries of quantum-inspired classical computing.

However, the limitations of today's noisy intermediate-scale quantum devices are evident. We are beginning to see real-world applications in areas like physics simulations, machine learning and optimization, but NISQ systems are still in the early stages of development and remain constrained by their sensitivity to error. As a result, quantum software today heavily relies on classical components for tasks like data management, pre-processing, results interpretation and noise mitigation, while quantum components handle specific, highly complex computational problems. NISQ devices hold scientific value, but no commercial application with demonstrated quantum advantage over classical methods has yet been identified, and are there no strong theoretical indications that such applications will emerge without fault-tolerant quantum computing[14].

There is a lack of mature development environments, standardised tools and high-level programming languages to make quantum software easy to develop and maintain. Although modular architectures and automation tools are slowly being introduced, much of the current focus is still on making quantum computing practical and accessible.

[13] Dalzell, A. M., McArdle, S., Berta, M., Bienias, P., Chen, C. F., Gilyén, A., ... & Brandão, F. G. (2023) 'Quantum algorithms: A survey of applications and end-to-end complexities', arXiv preprint arXiv:2310.03011.
[14] Paraphrased from John Preskill's talk at the US QIS Summer School Oak Ridge National Laboratory on 15 July 2024.

Stabilising quantum software currently revolves around optimising systems for NISQ hardware. The inherent noise, limited coherence times, and gate fidelities of NISQ systems mean that quantum software must evolve to address these challenges, integrating error mitigation techniques and automated quantum-classical orchestration tools that can handle the fluctuations of NISQ device performance. Modular architectures that combine quantum and classical components will be crucial to this stabilisation, allowing each system to perform the tasks they are best suited for.

To support scalable development there is a need for standardised frameworks and interoperable programming environments, reducing the complexities caused by the variety of quantum hardware. There must also be a focus on building verification and validation frameworks which can handle the probabilistic outcomes of quantum computations and provide reliable testing methodologies, ensuring the ecosystem is robust enough to support industry needs until fault-tolerant quantum computing becomes a reality.

Quantum software will evolve into complex (hybrid) systems that perfectly integrate quantum and classical components, able to tackle complex computational problems that are currently beyond the reach of classical computers alone. These systems will be designed to take advantage of the complementary strengths of both paradigms: quantum computers will handle tasks such as optimisation, or simulation of complex systems, while classical computers will perform data management, preprocessing and interpretation of results. Quantum software will likely be structured around modular architectures, in which quantum and classical modules work together, with high-level abstractions and orchestration layers managing the flow of information between the two. This hybrid approach will provide the flexibility needed to address real-world problems, making quantum computing a practical tool in finance, pharmaceuticals, logistics, and many other domains, perhaps drawing inspiration from the way GPUs can be programmed and used.

A key feature of future quantum software will be the presence of high-level programming languages which abstract much of the complexity of quantum mechanics, allowing developers to focus on problem solving rather than low-level quantum circuit design and sophisticated management of different types of hardware. These languages will likely resemble modern classical languages, equipped with quantum-specific constructs and libraries, allowing developers to define algorithms using familiar programming paradigms and even integrate with existing ones. In addition, automated quantum-classical orchestration tools will play an important role in closing the gap between quantum and classical resources, enabling their seamless integration. This will include sophisticated compilers which can optimise and map high-level code to specific quantum hardware architectures, considering hardware constraints such as the maximum number of qubits, maximum number of shots, maximum number of tasks, etc., and gate fidelity and decoherence times.

However, realising this vision presents numerous technical, methodological and practical challenges. The inherent nature of quantum mechanics radically changes the way software is developed, tested, and executed. Quantum systems do not follow classical logic, but are based on probabilistic results, superposition and entanglement, which makes algorithm design, error handling and verification particularly difficult. This complexity is compounded by the sensitivity of today's quantum hardware, leading to frequent computational errors. Developing effective error mitigation and correction techniques is therefore critical, but this adds complexity to software development. Current quantum computers remain limited in terms of the number of qubits and their ability to maintain coherence long enough to perform complex computations, making it difficult to create reliable large-scale quantum applications. Overcoming these limitations will require advances in both hardware and software.

Another significant challenge is the absence of standards for quantum software engineering and the lack of mature development tools. Unlike classical software, which has established ecosystems of integrated development environments (IDEs), libraries, testing frameworks and debugging tools, quantum computing is still in its early stages and lacks support in these areas. Each quantum processing provider offers its services

via its own language, libraries and programming abstractions. This significantly hinders productivity and increases the learning curve for new developers.

Quantum computing offers opportunities to advance classical software engineering practices, creating a synergistic relationship between the two fields. The potential of quantum computing to approximately solve large combinatorial optimization problems more efficiently could significantly improve classical software testing, verification and analysis processes. Quantum algorithms could be used to find optimal test paths through a large software code base, enabling faster identification of bugs and vulnerabilities. Quantum computing could provide new ways to simulate complex software systems, allowing software engineers to model and analyse scenarios that are currently too computationally intensive for classical methods. Tools and techniques developed for quantum software engineering are likely to be fed back into classical software practices, leading to innovations in the way we design, test and optimise software.

In conclusion, the future of quantum software is likely to be shaped by the convergence of quantum and classical computing, supported by the development of high-level languages, automated orchestration tools, new testing and verification methodologies, and the delivery of QaaS via the cloud. Overcoming the challenges posed by hardware limitations, complexity, and tooling shortcomings is critical to realising this vision. As these hurdles are overcome, quantum computing has the potential to evolve into a more practical and powerful tool, enabling applications beyond the reach of classical computing.

Editors

Josef Urban, Nokia;
Robert Wille, SCCH and Technical University of Munich;
Franz G. Fuchs, SINTEF; and
Jose Garcia-Alonso, University of Extremadura.