Exploring the Challenges of Hybrid Software with Quantum Design Patterns

Miriam Fernández-Osuna[a] and Ricardo Pérez-Castillo[b]
Faculty of Social Sciences & IT, University of Castilla-La Mancha, Talavera de la Reina, Spain

Keywords: Quantum Computing, Quantum Design Pattern, Quantum Software Architectures, Architecture Smells.

Abstract: Quantum computing has emerged as a new paradigm to solve several complex problems that are intractable for classical computers. This technology is being applied in areas such as optimization and cybersecurity, but its integration with classical software presents several challenges. One approach to overcome these obstacles is the adoption of design patterns, like those used in classical software, which could improve the scalability and maintainability of quantum systems. However, there is still a need to formalize architectural patterns that support this integration. Furthermore, quantum-classical software design can lead to problems that affect its quality, which highlights the importance of detecting and correcting them in time. This study presents an analysis and discussion of the challenges faced by hybrid software architectures such as problem modelling as well as dynamic generation of quantum circuits, execution orchestration, problem partitioning and interpretation of quantum results. The study of these challenges will serve as a starting point for proposing design patterns for hybrid software architectures.

1 INTRODUCTION

Quantum computing has emerged as a new computational paradigm that harnesses the principles of quantum mechanics, such as superposition and entanglement, to tackle algorithmically intractable problems in classical computing according to (Alsalman 2023). (Brookshear 1989) points out that thanks to its ability to simultaneously manipulate multiple quantum states, this technology has begun to be applied in sectors such as optimization, biomedicine, and cybersecurity, allowing NP-hard and NP-complete problems to be solved with unprecedented efficiency. However, the development of quantum software presents multiple challenges, especially when seeking its integration with classical software, giving rise to so-called hybrid software systems.

As mentioned by the Software Engineering Institute (SEI) (Carleton, Harper et al. 2021), the construction of these hybrid systems introduces significant challenges in architectural design, interoperability or the reuse of quantum components in classical architectures, among others.

One of the most promising approaches to improve hybrid software design and integration is the application of quantum design patterns (Leymann 2019). In classical software engineering, design patterns have proven to be an effective strategy for improving the maintainability and scalability of systems, and their application in quantum software could provide similar benefits (Gamma, Helm et al. 1993). The usage of design pattern allows the application of well-proven solutions for some of the common problems in hybrid software architectures. However, (Khan, Ahmad et al. 2023) explain that most documented patterns have focused on low-levels, such as quantum circuits and oracles, leaving a gap in the formalization of high-level architectural patterns that facilitate the effective integration of quantum software into existing systems.

In addition to design pattern reuse, another critical problem in building hybrid software according to (Khan, Ahmad et al. 2023) is the emergence of architecture smells, i.e. deficiencies in the architectural design that can compromise the quality and sustainability of the system over time. These problems may arise due to poor decisions in the

[a] https://orcid.org/0009-0006-8697-3816
[b] https://orcid.org/0000-0002-9271-3184

Fernández-Osuna, M. and Pérez-Castillo, R.
Exploring the Challenges of Hybrid Software with Quantum Design Patterns.
DOI: 10.5220/0013561100004525
In *Proceedings of the 1st International Conference on Quantum Software (IQSOFT 2025)*, pages 146-153
ISBN: 978-989-758-761-0

selection of quantum hardware, difficulties in the orchestration of interactions between classical and quantum systems, or the lack of reusability and adaptability of quantum circuits as its mentioned in (Pérez-Castillo, Serrano et al. 2021). Early detection and mitigation of these problems is crucial to ensure the efficiency and maintainability of hybrid software architectures.

This position paper presents and discusses some of the most common challenges present in hybrid software architectures. The main contribution is a critical analysis of those challenges and how these could be potentially addressed by the application of some of the existing classical/quantum design patterns, as well as motivate the need for further architectural patterns for hybrid software architectures. The integration of these systems in classical architectures is not trivial and requires solving technical challenges: Problem modelling, dynamic generation of quantum circuits, execution orchestration, problem partitioning, and the interpretation of the quantum results.

The structure of the paper is as follows: Section 2 presents the state of the art and related work; Section 3 describes all the challenges faced by the hybrid architecture; Section 4 discusses the main implications for researchers and practitioners and Section 5, explains the conclusions and future work.

2 STATE-OF-THE-ART

This section presents the state of the art of quantum computing and quantum software (cf. Section 2.1) and quantum design patterns (cf. Section 2.2).

2.1 Quantum Computing and Quantum Software

As discussed in section 1, quantum computing surpasses classical computing in solving complex problems. The qubit, its fundamental unit, can be implemented through various physical systems (Ezratty 2021). Quantum supremacy has already been demonstrated (Arute, Arya et al. 2019). The field is advancing rapidly, IBM introduced a processor in 2021 (Chow, Dial et al. 2021) and announced the Condor in 2023 (Gambetta 2023). Google predicts a million-qubit machine by the decade's end.

Quantum computing tackles NP-hard and NP-complete problems in optimization, cryptography, and machine learning, outperforming classical methods for large instances (Brookshear 1989). (Ukpabi, Karjaluoto et al. 2023), highlights its transformative impact across industries.

Despite its potential, the NISQ era faces challenges like error correction and scalability (Preskill 2018). However, advantages may emerge even before full fault tolerance (Kim, Eddins et al. 2023). Beyond hardware, quantum software is essential to unlock quantum computing's full power (Piattini, Peterssen et al. 2020).

The *Quantum Software Manifesto*, proposed by (Jiménez-Navajas, Pérez-Castillo et al. 2025), stresses the importance of advancing in this field, while the *European Quantum Flagship*, according to (Serrano, Cruz-Lemus et al. 2022), proposes its integration with classical systems. As (Rieffel and Polak 2011) point out, unlike classical software, quantum software uses probabilistic functions and quantum gates to manipulate cubits.

There are several quantum programming languages, such as OpenQASM, Q#, Qiskit, Cirq and pyQuil, which have significant differences, such as the absence of traditional control structures and the need to define parameterizable programs. In addition, most quantum development platforms, such as IBM Q Experience, Amazon Bracket and D-Wave Leap, operate in the cloud, allowing both simulations and execution on real hardware.

However, quantum software is not suitable for all tasks due to its costs and limitations. Therefore, previous studies such as (Pérez-Castillo, Serrano et al. 2021) have explored hybrid systems, in which classical software manages quantum execution and result processing. These systems must address challenges such as code integration, validation, and portability, applying software engineering principles such as abstraction, automation, and quality assurance.

Hybrid software, which integrates classical and quantum computing, plays a crucial role in facilitating the adoption of quantum technologies in real-world applications according to (Garcia-Alonso, Rojo et al. 2022). Quantum computers, typically accessed through cloud-based APIs (e.g., IBM, D-Wave), perform computations on problems specified by classical systems and return probabilistic results, which require classical post-processing. This interaction introduces challenges in selecting appropriate quantum components, managing interoperability, and optimizing execution performance (Carleton, Harper et al. 2021).

The design of hybrid systems must address several architectural challenges, including the selection of classical or quantum solutions based on functional requirements, orchestration of quantum

invocations, and efficient error handling and result interpretation. Additionally, most quantum circuits are dynamically built during execution, necessitating adaptive integration mechanisms.

2.2 Quantum Design Pattern

Research on design patterns in quantum software has evolved from initial approaches focusing on circuit-level patterns to address architectural questions of greater complexity. As (Weigold, Barzen et al. 2021) point out, early studies focused on measurement patterns within one-way quantum computing, but the field has progressed considerably.

A significant breakthrough was the proposal of a pattern language for quantum algorithms in the work of (Khan, Ahmad et al. 2023), where fundamental patterns such as initialization and uniform superposition were established. These were later extended with techniques for data encoding, error handling and hybrid quantum-classical integration (Baczyk, Pérez-Castillo et al. 2024).

Today, (Beisel 2025) has documented catalogues of software patterns designed to optimize quantum circuit design. Ten essential patterns are identified, including entanglement creation, oracle, amplitude amplification and quantum-classical splitting. These patterns exploit unique properties of quantum states to improve efficiency and optimize resource use.

Unfortunately, all these related proposals do not address specific challenges for defining hybrid software architectures at a high level of abstraction.

3 CHALLENGES IN HYBRID SOFTWARE ARQUITECTURES

This section presents five challenges that we consider to be the most recurrent and impactful in the context of hybrid software architectures: problem modelling (cf. Section 3.1), dynamic generation of quantum circuits (cf. Section 3.2), execution orchestration (cf. Section 3.3), problem partitioning (cf. Section 3.4) and interpretation of quantum result (cf. Section 3.5). These challenges were selected based on their frequent appearance in the literature on quantum-classical hybrid systems, their centrality to the development of effective software pipelines, and their significant influence on the overall performance and scalability of hybrid solutions. While other challenges exist, the selected ones are those that most directly affect the architectural design of hybrid applications and their software engineering processes.

3.1 Problem Modelling

First, the certain quantum software algorithm should be chosen to solve a given problem. Usually, the algorithm selection is a fix decision at design time. However, sometimes, this decision could be made, between some valid algorithms, at runtime, for example depending on the complexity and size of the input.

Let us imagine that we try to optimize a logistic route between some cities, i.e., the Traveling Salesman Problem (TSP). The hybrid software architecture could incorporate mechanisms to determine when the Quantum Approximate Optimization Algorithm (QAOA) (Blekos, Brand et al. 2024), commonly used for route optimization, is unnecessary. This would apply in cases where the input, which varies overtime, involves only a small number of cities, allowing for the use of classical software instead.

Thus, the proper problem modelling and its associated data preparation entails a relevant challenge for hybrid software architectures. This task also affects the dynamic generation of parameterized quantum algorithms with the input data, usually managed by classical counterpart. This data can come from various sources, be heterogeneous and constantly changing, which requires adequate pre-processing prior to its use in quantum computing. Since this pre-processing is managed by classical software, it has a direct impact on the configuration and execution of quantum circuits, and its correct structuring is essential to optimize the performance of the hybrid system.

Moreover, this issue not only influences the dynamic generation of circuits, but also the definition of parameters that guide the execution of quantum algorithms in a self-adaptive way. For example, depending on certain characteristics of the quantum hardware or problem-specific constraints, optimization parameters can be dynamically adjusted to improve the efficiency of the execution (Sepúlveda, Pérez-Castillo et al. 2025).

In this context, problem modelling encompasses both the transformation and fitness of the input data as well as the optimal configuration of the quantum algorithm. All these tasks should be properly designed in the target hybrid software architecture.

3.2 Dynamic Generation of Quantum Circuits

Depending on the problem modelling, and dynamically changing input data, quantum circuits

for solving the specific problem must be built at runtime (Quetschlich, Burgholzer et al. 2023). This poses another interesting challenge for hybrid software architecture.

This challenge directly affects the transparency and predictability of the software design, complicating its maintainability and scalability. In hybrid architectures, where quantum and classical computing must be efficiently integrated, the lack of clear rules in the dynamic generation of circuits introduces uncertainty in the behavior of the system.

Design patterns should be applied in this context to establish a well-defined structure for circuit generation, ensuring that the rules are explicit and easily traceable from the problem definition. By providing some architectural solutions for circuit creation and modification, the traceability of the execution flow is improved, facilitating debugging and maintenance.

A more structured system is achieved where each modification in the circuit responds to well-defined criteria. This ensures that changes remain within the same compilation unit (such as a class), facilitating the software's evolution over time. It also allows the design to be understandable and predictable, which favors the optimization of resources and the integration with classical algorithms in a more efficient way. In addition, the reuse of components within the system is facilitated, improving the scalability of hybrid software. However, it is important to note that in this article, we do not apply or propose design patterns; rather, we discuss the challenges associated with their use in quantum software, focusing on the difficulties in designing structured and scalable systems rather than advocating for specific pattern implementations.

3.3 Execution Orchestration

A sophisticated hybrid software system can efficiently manage multiple quantum algorithms to address a range of related computational problems. The selection of an appropriate algorithm depends not only on the nature and scale of the problem, as previously discussed, but also on the optimal allocation of quantum computing resources, considering hardware constraints (Weder, Barzen et al. 2021). This approach ensures efficient utilization of available quantum hardware while adapting to the specific requirements of each problem instance.

This challenge impact the hybrid software architecture. For example, multiple quantum circuits that are managed by the same controller is clearly an inefficient solution. One of the main problems is excessive coupling. When a single controller is responsible for coordinating the execution of more than one quantum circuit, it becomes a critical node. Any failure in its operation can compromise the system, affecting the execution of computations and the correct allocation of resources. In addition, this structure limits scalability. As more circuits are added, the controller's overhead increases, reducing its ability to efficiently distribute tasks.

Another important aspect is the difficulty in managing quantum resources. Each circuit may require specific quantum gate configurations, qubit optimization or error correction strategies. A centralized controller struggles to manage this diversity effectively, leading to a decline in overall performance. Also, managing quantum states can become complex. If multiple circuits rely on a single controller to manage result retrieval, interference and loss of computational consistency can occur.

To mitigate these problems, a design pattern based on controller distribution can be applied. Instead of a single controller for multiple circuits, each quantum circuit has its own independent controller. This allows for a more accurate allocation of resources and prevents a failure from affecting the entire system. It also improves scalability, as the addition of new circuits does not overload a single central unit. Flexibility is also enhanced, as circuits can be configured more precisely without one affecting the others.

3.4 Problem Partitioning

Problem partitioning is a fundamental challenge that directly affects the feasibility of the solution implemented in a hybrid software architecture. This challenge arises due to differences in computing paradigms: while classical systems are suitable for sequential and general-purpose operations, quantum systems offer significant advantages in specific problems, such as optimization, simulation of physical systems and complex data analysis. However, quantum resources are constrained in terms of number of qubits, coherence time and logic gate fidelity, which requires careful structuring of how tasks are delegated between the two paradigms.

For example, one of the limitations in this regard is the lack of enough qubits. As a solution, some proposals (Ariño Sales and Palacios Araos 2023) have divided the whole problem into smaller ones, e.g., by classical clustering, that can be then solved independently on quantum hardware that fulfil some constraints. This entails two additional challenges. First, the input data preprocessing (cf. section 3.1),

which, in many cases, should oversee the problem partition to fit the capabilities of the quantum system. Second, the output post-processing for combining individual outputs and providing a relevant solution for the whole problem (cf. section 3.5).

Appropriate partitioning seeks to maximize the utilization of quantum computing without overloading it with processes that can be solved more efficiently in the classical part of the system. This involves designing strategies to divide the problem into manageable subproblems and distributing them according to the capabilities of each environment. Incorrect partitioning can lead to unnecessary redundancies, loss of information in the communication between the two parts and, in the worst case, make the execution of a hybrid algorithm unfeasible.

As a result, there could be the need for some design pattern that address this challenge by providing a structured framework for defining how the problem should be optimally partitioned. It focuses on criteria such as the computational complexity of each part, the capacity of available quantum devices and the costs associated with transferring data between classical and quantum components. By applying this pattern, it achieves greater effectiveness in hybrid software execution, reducing latency and improving overall system performance. It also facilitates more effective scalability by allowing the partitioning to adapt as quantum resources evolve and capacity increases.

Beyond performance, this approach has implications for the correctness and stability of hybrid algorithms, since a correct distribution of tasks avoids the accumulation of quantum errors and optimizes the use of noise mitigation techniques. Thus, the design pattern not only solves the partitioning problem, but also contributes to the robustness and reliability of hybrid solutions, ensuring that the interaction between classical and quantum systems is smooth and effective.

3.5 Interpretation of Quantum Results

The challenge in hybrid software architectures refers to the need of interpreting quantum results effectively, provide meaningful solutions for the end users. This is specifically relevant when problem partitioning happens (see section 3.4). This implicates a spare post-processing. In these environments, the core issue lies in how the problem is modelled, which directly impacts the clarity and usability of the results. The execution of various functions and methods to process a solution generates outputs that, without a structured interpretation layer, can be difficult to understand and apply in practice.

This lack of a coherent interpretative structure is particularly problematic in hybrid systems, where quantum and classical computations produce data in different formats or levels of abstraction. Without a proper framework for interpretation, users may struggle to assess the reliability of the results and make informed decisions. Thus, the problem is not just handling fragmented outputs but ensuring that the results are presented in a way that makes sense within the broader context of the modelled problem (explained in section 3.1).

In classical software architectures, the Model-View-Controller (MVC) by (Gamma, Helm et al. 1993) design pattern has been successfully applied. MVC enhances modularity, maintainability, and scalability by separating an application into three components: the Model, which encapsulates the core logic, including quantum algorithms and interactions with quantum hardware; the View, responsible for displaying information in a user-friendly manner, such as visualizing quantum results or circuit representations; and the Controller, which mediates user input, orchestrates hybrid quantum-classical processing, and ensures appropriate data flow. In hybrid software architecture, MVC plays a crucial role by abstracting quantum complexity within the Model, allowing the Controller to select appropriate quantum resources and oversee post-processing of probabilistic results, and enabling the View to present meaningful interpretations to users. However, applying this pattern becomes more complex due to the stochastic nature of quantum software, which requires multiple shots to obtain meaningful solutions, as measurements are derived from probability distributions rather than deterministic outputs. Additionally, quantum jobs are often queued, meaning the system must efficiently manage job execution times, track when responses are obtained, and synchronize results with classical processing. This introduces challenges in managing asynchronous quantum requests and ensuring that users receive timely, interpretable solutions, reinforcing the need for a robust orchestration layer within the Controller to handle scheduling, retrieval, and integration of quantum results.

4 MAIN IMPLICATIONS

Hybrid software architectures pose challenges to efficiency, maintainability, and scalability, requiring structured strategies to integrate classical and quantum computing effectively. Section 4.1 summarizes the key

challenges while section 4.2 discusses their implications and the architectural needs.

4.1 Key Challenges in Hybrid Software Architectures

This section presents a summary of the challenges, main impacts in hybrid software architectures as well as main quality characteristics affected.

Sparse Pre-processing and Fragmented Data Handling

- **Challenge Summary:** In hybrid software architectures, classical data must be properly structured and encoded before being processed by quantum algorithms. However, current approaches often distribute pre-processing tasks across different layers or modules without a centralized strategy, leading to inconsistencies in data representation.
- **Impact:** This fragmented handling of data makes it difficult to track transformations, introduces latency, and complicates data integration with quantum computation. The lack of standardization also reduces software maintainability and interoperability with different quantum backends.
- **Quality characteristics affected:** Maintainability, interoperability, performance.

Unstructured Quantum Circuit Generation

- **Challenge Summary:** Many hybrid software systems generate quantum circuits dynamically at runtime based on input data. However, without predefined rules for structuring these circuits, the process becomes unpredictable, making debugging, performance optimization, and reproducibility difficult.
- **Impact:** The absence of structured circuit generation affects software maintainability, as dynamically generated circuits may differ between executions, reducing traceability. Additionally, inefficient circuit structures can lead to increased execution times and suboptimal use of quantum hardware.
- **Quality characteristics affected:** Maintainability, traceability, scalability.

Fragmented Post-processing and Result Interpretation

- **Challenge Summary:** Quantum computations typically produce probabilistic results that require post-processing before they can be used effectively in decision-making processes. When post-processing is distributed across multiple software components without a unified framework, integrating and interpreting results becomes challenging.
- **Impact:** Without a structured post-processing mechanism, quantum results may lack clarity, reducing their usability. Additionally, inconsistencies in result aggregation and interpretation may compromise the reliability of hybrid systems.
- **Quality characteristics affected:** Usability, reliability, interpretability.

Centralized Execution Orchestration

- **Challenge Summary:** Hybrid software often relies on a centralized controller to manage multiple quantum circuits or algorithms. While this approach simplifies initial development, it creates bottlenecks that affect the efficiency and scalability of the system.
- **Impact:** Centralizing execution orchestration increases system coupling, making it difficult to scale as the number of quantum tasks grows. A single point of failure also reduces system reliability and flexibility, as every execution request depends on a single coordination mechanism.
- **Quality characteristics affected:** Scalability, fault tolerance, flexibility.

Problem Partitioning Complexity

- **Challenge Summary:** Hybrid software requires partitioning computational tasks between classical and quantum components. However, determining which parts of a problem should be executed on quantum hardware versus classical processors is complex and depends on hardware constraints, algorithmic efficiency, and problem structure.
- **Impact:** Incorrect problem partitioning can lead to inefficiencies, excessive communication overhead, or infeasible hybrid execution. If quantum resources are not optimally utilized, classical components may end up handling most of the workload, reducing the benefits of quantum acceleration.
- **Quality characteristics affected:** Performance, adaptability, feasibility.

4.2 Architectural Needs for Hybrid Software Systems

To address these challenges, hybrid software architectures require structured design principles. Below, we outline key architectural needs that should be incorporated into future high-level design patterns:

1. **Pre-processing Standardization.** Establishing a centralized transformation mechanism ensures that input data follows a consistent format, improving traceability, maintainability, and interoperability with quantum systems. This includes defining common encoding schemes for quantum algorithms and integrating pre-processing steps into a modular framework to simplify data preparation.
2. **Structured Circuit Generation.** Defining clear rules and templates for dynamic quantum circuit generation ensures predictability and reproducibility. This includes using modular circuit components that follow a standardized approach to gate arrangement, qubit mapping, and circuit optimization techniques.
3. **Post-processing Interpretation Layer.** Implementing a structured post-processing framework centralizes result aggregation, filtering, and error mitigation, improving clarity and usability. This ensures that quantum results are transformed into meaningful outputs for classical decision-making, enhancing system interpretability and reliability.
4. **Distributed Execution Orchestration.** Introducing independent execution controllers for different quantum circuits reduces bottlenecks and improves system scalability. This approach enables parallel execution of multiple quantum tasks, increases system flexibility, and enhances fault tolerance by preventing failures in one component from affecting the entire execution pipeline.
5. **Partitioning Strategies for Hybrid Workloads.** Developing automated workload distribution strategies that analyze computational complexity and assign tasks to classical or quantum processors based on efficiency criteria. Ensuring that partitioning strategies dynamically adapt to quantum hardware capabilities and optimize data flow between classical and quantum components.

5 CONCLUSION AND FUTURE WORK

This study analyzes the challenges of hybrid software architectures, including problem modeling, dynamic quantum circuit generation, execution orchestration, problem partitioning, and the interpretation of quantum results. Addressing these challenges is essential for improving hybrid software quality, ensuring properties such as maintainability and scalability, and facilitating the broader adoption of quantum computing in industry.

This study has limitations, including potential subjectivity in the selection of challenges and the lack of empirical validation, which affects the generalizability of the results. Future work should address these issues and consider the evolving nature of quantum technologies. A key direction for future research is the identification of design and architectural patterns to systematically address these challenges. Defining high-level, reusable design patterns can support modular quantum software development, scalable quantum-classical integration, and efficient problem partitioning. Additionally, it is crucial to identify and mitigate architectural antipatterns—common but suboptimal design choices that may introduce or exacerbate the challenges discussed. By formalizing both effective design patterns and potential pitfalls, hybrid quantum software can become more robust, maintainable, and scalable, accelerating its practical adoption.

ACKNOWLEDGEMENTS

This work is supported by the projects SMOOTH (PID2022-137944NB-I00) and QU-ASAP (PDC2022-133051-I00) funded by MICIU/AEI/ 10.13039/501100011033 / PRTR, EU.

REFERENCES

Alsalman, A. I. S. (2023). "Accelerating Quantum Readiness for Sectors: Risk Management and Strategies for Sectors." Journal of Quantum Information Science **13**(2): 33-44.

Ariño Sales, J. F. and R. A. Palacios Araos (2023). "Adiabatic quantum computing impact on transport optimization in the last-mile scenario." Frontiers in Computer Science **5**: 1294564.

Arute, F., K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao and D. A. Buell (2019). "Quantum supremacy using a

programmable superconducting processor." Nature **574**(7779): 505-510.

Baczyk, M., R. Pérez-Castillo and M. Piattini (2024). Towards a Framework of Architectural Patterns for Quantum Software Engineering. 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE.

Beisel, M. a. B., Johanna and Leymann, Frank and Weder, Benjamin (2025). "Operations Patterns for Hybrid Quantum Applications."

Blekos, K., D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya and A. Summer (2024). "A review on quantum approximate optimization algorithm and its variants." Physics Reports **1068**: 1-66.

Brookshear, J. G. (1989). Theory of computation: formal languages, automata, and complexity, Benjamin-Cummings Publishing Co., Inc.

Carleton, A., E. Harper, J. E. Robert, M. H. Klein, D. De Niz, E. Desautels, J. B. Goodenough, C. Holland, I. Ozkaya and D. Schmidt (2021). "Architecting the future of software engineering: A national agenda for software engineering research and development." Softw. Eng. Inst., Pittsburgh, PA, USA, AD1152714.

Carleton, A. D., E. Harper, J. E. Robert, M. H. Klein, D. De Niz, E. Desautels, J. B. Goodenough, C. Holland, I. Ozkaya and D. Schmidt (2021). Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research and Development, Software Engineering Institute, Carnegie Mellon University.

Chow, J., O. Dial and J. Gambetta (2021). "IBM Quantum breaks the 100-qubit processor barrier." IBM Research Blog.

Ezratty, O. (2021). Understanding quantum technologies, le lab quantique.

Gambetta, J. (2023). The hardware and software for the era of quantum utility is here, IBM. https://www.ibm.com/quantum/blog/quantum-roadmap-2033.

Gamma, E., R. Helm, R. Johnson and J. Vlissides (1993). Design patterns: Abstraction and reuse of object-oriented design. ECOOP'93—Object-Oriented Programming: 7th European Conference Kaiserslautern, Germany, July 26–30, 1993 Proceedings 7, Springer.

Garcia-Alonso, J., J. Rojo, D. Valencia, E. Moguel, J. Berrocal and J. M. Murillo (2022). "Quantum Software as a Service Through a Quantum API Gateway." IEEE Internet Computing **26**(1): 34-41.

Jiménez-Navajas, L., R. Pérez-Castillo and M. Piattini (2025). "Transforming Quantum Programmes in KDM to Quantum Design Models in UML." Informatica: 1-42.

Khan, A. A., A. Ahmad, M. Waseem, P. Liang, M. Fahmideh, T. Mikkonen and P. Abrahamsson (2023). "Software architecture for quantum computing systems — A systematic review." Journal of Systems and Software **201**: 111682.

Kim, Y., A. Eddins, S. Anand, K. X. Wei, E. Van Den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel and K. Temme (2023). "Evidence for the utility of quantum computing before fault tolerance." Nature **618**(7965): 500-505.

Leymann, F. (2019). Towards a pattern language for quantum algorithms. Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings 1, Springer.

Piattini, M., G. Peterssen, R. Pérez-Castillo, J. L. Hevia, M. A. Serrano, G. Hernández, I. G. R. De Guzmán, C. A. Paradela, M. Polo and E. Murina (2020). The talavera manifesto for quantum software engineering and programming. QANSWER.

Preskill, J. (2018). "Quantum Computing in the NISQ era and beyond." Quantum **2**(aug): 79.

Pérez-Castillo, R., M. A. Serrano and M. Piattini (2021). "Software modernization to embrace quantum technology." Advances in Engineering Software **151**: 102933.

Quetschlich, N., L. Burgholzer and R. Wille (2023). Reducing the Compilation Time of Quantum Circuits Using Pre-Compilation on the Gate Level. 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE Computer Society**:** 757-767.

Rieffel, E. G. and W. H. Polak (2011). Quantum computing: A gentle introduction, MIT press.

Sepúlveda, S., R. Pérez-Castillo and M. Piattini (2025). "A software product line approach for developing hybrid software systems." Information and Software Technology **178**: 107625.

Serrano, M. A., J. A. Cruz-Lemus, R. Perez-Castillo and M. Piattini (2022). "Quantum software components and platforms: Overview and quality assessment." ACM Computing Surveys **55**(8): 1-31.

Ukpabi, D., H. Karjaluoto, A. Bötticher, A. Nikiforova, D. Petrescu, P. Schindler, V. Valtenbergs and L. Lehmann (2023). "Framework for understanding quantum computing use cases from a multidisciplinary perspective and future research directions." Futures **154**: 103277.

Weder, B., J. Barzen, F. Leymann and M. Zimmermann (2021). Hybrid quantum applications need two orchestrations in superposition: a software architecture perspective. 2021 IEEE International Conference on Web Services (ICWS), IEEE.

Weigold, M., J. Barzen, F. Leymann and D. Vietz (2021). Patterns for hybrid quantum algorithms. Symposium and Summer School on Service-Oriented Computing, Springer.