Unraveling Quantum Computing System Architectures: An Extensive Survey of Cutting-Edge Paradigms

Xudong Zhao[a], Xiaolong Xu[a,*], Lianyong Qi[b,*], Xiaoyu Xia[c], Muhammad Bilal[d], Wenwen Gong[e], Huaizhen Kou[f]

[a]*School of Software, Nanjing University of Information Science and Technology, Nanjing, China*
[b]*College of Computer Science and Technology, China University of Petroleum (East China), Qingdao, China*
[c]*School of Computing Technologies, RMIT University, Melbourne, Victoria, Australia*
[d]*School of Computing and Communications, Lancaster University, Bailrigg, Lancaster LA1 4WA, United Kingdom*
[e]*Department of Computer Science and Technology, Tsinghua University, Beijing, China*
[f]*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China*

Abstract

Context: The convergence of physics and computer science in the realm of quantum computing systems has sparked a profound revolution within the computer industry. However, despite such promise, the existing focus on quantum software systems primarily centers on the generation of quantum source code, inadvertently overlooking the pivotal role of the overall software architecture.

Objectives: In order to provide comprehensive guidance to researchers and practitioners engaged in quantum software development, employing an architecture-centered development model, an extensive literature review was conducted pertaining to existing research on quantum software architecture. The analysis encompasses a detailed examination of the characteristics exhibited by these studies and the identification of prospective challenges that lie ahead in the field of quantum software architecture.

Methods: We have closely examined instances of quantum software en-

*Corresponding author
Email address: njuxlxu@gmail.com (Xiaolong Xu), lianyongqi@gmail.com (Lianyong Qi)

gineering, quantum modeling languages, quantum design patterns, and quantum communication security to gain insights into the distinctive attributes associated with various software architecture approaches.

Results: Our findings underscore the critical significance of prioritizing software architecture in the development of robust and efficient quantum software systems. Through the synthesis of these multifaceted aspects, both researchers and practitioners can devise quantum software solutions that are inherently architecture-centric.

Conclusion: The software architecture of quantum computing systems plays a pivotal role in determining their ultimate success and usability. Given the ongoing advancements in quantum computing technology, the migration of traditional software architecture development methods to the domain of quantum software development holds significant importance.

Keywords: Quantum Computing, Quantum Software Architecture, Quantum Software Engineering.

---

1. Introduction

Quantum computing, along with the broader field of quantum information, encompasses a collection of concepts and technical systems that delve into the nature of information and its processing, grounded in the principles of quantum mechanics [1]. The evolution of quantum computing ideas and concepts has undergone a relatively protracted period, during which physicists and mathematicians have played pivotal roles in advancing the development of quantum computing. The developmental trajectory of quantum computing can be divided into the following epochs [2]:

- Theoretical Era (Early 1980s to Early 1990s): During this period, the theoretical foundation of quantum computing was established. In 1982, Feynman introduced the concept of quantum computing, followed by the formulation of basic principles of quantum algorithms by Deutsch, Bennett, and others. In 1994, Peter Shor proposed the renowned Shor's algorithm [3], demonstrating that quantum computing can solve NP problems in polynomial time. The hallmark of this era was a predominance of theoretical investigations, as experimental techniques were not yet fully mature.

- Experimental Era (Mid-1990s to Mid-2000s): In this era, experimental techniques in quantum computing experienced rapid advancement. In 1995, IBM Laboratory successfully demonstrated quantum computing with two qubits. In 1998, Los Alamos National Laboratory achieved quantum computing with seven qubits. This period was characterized by swift progress in experimental techniques, though the number of qubits remained quite limited.

- Engineering Era (Mid-2000s to Present): In this period, quantum computing entered the engineering phase. In 2007, IBM Laboratory unveiled the first commercially available quantum computer. In 2016, Google Laboratory announced the achievement of quantum supremacy. This era is marked by the commercialization and practical application of quantum computers, accompanied by substantial enhancements in the quantity and quality of qubits.

In recent years, with the advent of quantum algorithms, Quantum Programming Languages (QPLs), and quantum compilers, programmers have been able to use the theories and principles of quantum mechanics to process information and perform specific computational tasks at a much higher speed compared to classical computer systems [4, 5, 6, 7]. In recent research endeavors, a substantial number of scholars have amalgamated the distinctive attributes of quantum computing with algorithms in various domains, in a concerted effort to collectively enhance the efficacy of these algorithms. Edge computing [8], as an emerging computational paradigm, achieves lower latency and higher efficiency by deploying computational tasks in close proximity to data sources. When coupled with quantum computing, it further augments the performance of edge computing, particularly in scenarios necessitating efficient processing of extensive datasets, thereby endowing quantum computing with substantial support for edge computing endeavors. Machine learning [9] and data mining [10], domains reliant on extensive data processing and analysis, stand to benefit profoundly from quantum computing. The parallel computational capabilities inherent to quantum computing render it an ideal choice for handling intricate machine-learning models and large-scale data mining tasks. Through adept utilization of quantum algorithms, researchers can expedite training processes and optimize model performance, thereby propelling advancements in the realms of machine learning and data mining. Cloud computing [11], characterized by network-based computational models, has emerged as a primary provisioning mechanism for computing resources in numerous enterprises and organizations. Infusing quantum computing into cloud computing architectures offers users substantially augmented computational capabilities, thereby supporting complex scientific computations, simulations, and data processing tasks. This fusion also charts a new trajectory for the evolution of cloud services, driving further enhancements in cloud-based functionalities.

In contrast to classical computers, quantum computers possess the remarkable ability to concurrently handle multiple computational tasks and execute complex computations swiftly. However, the current focus of QPLs and their corresponding algorithms primarily revolves around computational and implementation specifics, aiming to generate executable specifications. This narrow perspective tends to overlook the comprehensive global view of the software system under design. By excessively emphasizing source code implementation details, the architectural viewpoint, which serves as a blueprint for the system, is undermined, potentially compromising the quality and functionality of the final product, namely quantum software. Nevertheless, leading technology companies are significantly increasing their financial and strategic investments in quantum computing platforms, with a particular emphasis on QPLs, such as Microsoft's Q# [12], IBM's Qiskit [13], and Google's Cirq. It is worth noting that the field of quantum software engineering is still in its nascent stage. Recent research has highlighted that quantum software projects, which neglect fundamental design principles and prioritize the implementation of quantum source code, often yield suboptimal and error-prone outcomes. For instance, with regard to performance concerns, non-optimized code may engender suboptimal execution times, notably when tasked with computations of substantial scale within the domain of quantum computing. Such inefficiencies may culminate in instances of timeout or inadequacy in resource allocation, thereby impeding the successful completion of computational tasks [14]. Pertaining to modularity and the promotion of code reusability, a dearth in modular design engenders verbosity, augments maintenance complexity, and obfuscates comprehensibility. Consequently, during the process of function extension or modification, the propensity for inadvertent error introduction or unforeseen behavioral outcomes is notably heightened.

To invigorate the intellectual discourse and scholarly investigations within the academic community, and to furnish a comprehensive standpoint on Quantum Software Architecture (QSA) research, this paper undertakes a systematic review encompassing the definition and research methodologies pertaining to quantum computing software architecture. The second section of this paper elucidates the distinctive attributes of quantum computing, quantum software engineering, and the precise definition of quantum computing software architecture. Subsequently, the third section expounds upon the specific applications, directions, and role of QSA. In the fourth section, the research emphasis on constructing software architecture was introduced, encompassing the exploration of Q-UML, architecture design patterns, and quantum communication technology. Lastly, the fourth section consolidates recent research findings in this domain, while offering recommendations and insights regarding future developmental trajectories.

2. Basic Concepts

In this section, we provide a brief review of the concepts in quantum computing and software architecture.

2.1. Quantum Computing VS Classical Computing

Quantum computing is a product of the combination of quantum mechanics and computer science. According to Moore's law, the number of transistors integrated into a chip grows exponentially with time. When the storage unit of a computer reaches the atomic scale, significant quantum effects will seriously affect its performance. These quantum effects encompass phenomena such as quantum tunneling, superposition, and entanglement, which manifest prominently at the quantum level.

Quantum tunneling involves particles traversing energy barriers that classical physics would deem impassable. Superposition allows qubits to exist in multiple states simultaneously, enabling quantum computers to explore diverse possibilities concurrently. Entanglement establishes a correlation between quantum systems, leading to instantaneous connections across vast distances. As the storage units shrink toward atomic dimensions, these quantum effects become increasingly influential, posing substantial hurdles for conventional computers.

This is the fundamental difficulty encountered in the development of conventional computers [15]. Further development of computer science (CS) requires the help of new principles and methods, and quantum computing provides a new way to solve this problem.

Similarly, the basic unit of information storage and processing in quantum computing is the qubit, which is in a quantum superposition of states $|0\rangle$ and $|1\rangle$.

As stated in Heisenberg's uncertainty principle, "the position of a moving particle and its momentum cannot be determined simultaneously", we cannot get the exact state of a quantum. According to the theorem of physics, once we measure or observe the quantum (position, momentum, etc.), it will inevitably lead to the collapse of the quantum state. We can only get the state determined after the measurement, but we cannot confirm the previous state of the quantum. The state of a quantum with state $|\psi\rangle$ can be expressed by

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \tag{1}$$

and

$$|\alpha_0|^2 + |\alpha_1|^2 = 1, \tag{2}$$

where α_0 is the square of the probability that the quantum will get 0 after measurement, and α_1 is the square of the possibility that the quantum will get 1 after measurement.

Eq.(1) and Eq.(2) solely delineate the superposition equation governing the state in the context of a single bit. In the scenario of two bits, the state's superposition equation is expressed as indicated in Eq.(3).

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle \tag{3}$$

In summary, the most prominent distinction between qubits and classical bits lies in their state space and the nature of permissible operations. The capacity of qubits to exist in superposition states and undergo quantum gate operations confers quantum computing with advantages over classical computing in specific tasks, particularly when addressing complex problems.

2.2. Quantum Software Engineering (QSE)

Since the late 1960s, scholars in the field of computer science have been actively addressing the challenges posed by the "Software Crisis", a term denoting persistent challenges and inherent deficiencies within the software development process. In response to this enduring predicament, researchers have diligently explored a diverse array of techniques and methodologies aimed at alleviating these sustained issues, as substantiated by the comprehensive study conducted in [16]. In recent years, with the continuous development of science and technology and the essence of social development needs, software engineering gradually towards the road of intelligence, information technology, and integration, and the emergence of new software engineering models such as the Extreme Programming Model (XP model), component-based models and other new software engineering models. These models improve the efficiency and quality of software development by improving the logic flow of the model and the reusability of the code.

The term "quantum software engineering" was originally coined by John Clark et al. [17] at the 2002 workshop on Grand Challenges in Computing Research. Since then, quantum software engineering has been studied in depth, with many papers discussing the challenges and opportunities of quantum computing in the quantum software development process.

Stepney et al. [18] presented a pivotal challenge in the realm of quantum software engineering: the establishment of a mature discipline for quantum software engineering, aimed at fully leveraging the potential of commercial quantum computer hardware. They posit that "the entire classical software engineering needs to be redesigned and extended to the quantum domain" and delineate this challenge from multiple perspectives of quantum software engineering, encompassing quantum computing models, languages, quantum compilers, methods, and tools.

Certainly, the software engineering and quantum computing communities have focused on the above challenges to contribute to the overall construction of the quantum software engineering field. Barbosa et al. [19] outlined several research challenges that must be addressed to advance software engineering applications in quantum computing. For instance, most current quantum algorithms assume the availability of a large number of qubits capable of indefinitely preserving information, a technological capability not presently attained. Ahmad et al. [20], on the other hand, highlighted the complexities inherent in quantum computing systems and technology development, encompassing challenges associated with quantum computer programming, operation, and maintenance, as well as the distinct engineering paradigms involved. Many researchers have used the classical SE process for the design and development of quantum software. For instance, UML use case diagrams can depict external entities and use cases within a quantum software system. Researchers, as delineated in [21], concentrate on the application of agile practices in quantum software development, endeavoring to address associated challenges and limitations.

Since quantum computing technology is still in its infancy, we still do not have clear models, standards, or methods to help us create new systems and migrate existing ones. Considering the current state of quantum computing, we need to go back to the path taken by software engineering in the last century to achieve a new golden age of QSE [22].

2.3. Quantum Software Architecture (QSA)

While the scale of software systems is rapidly increasing, software development methods have undergone a series of changes. In the process, software architecture has evolved from a vague concept to a maturing technology.

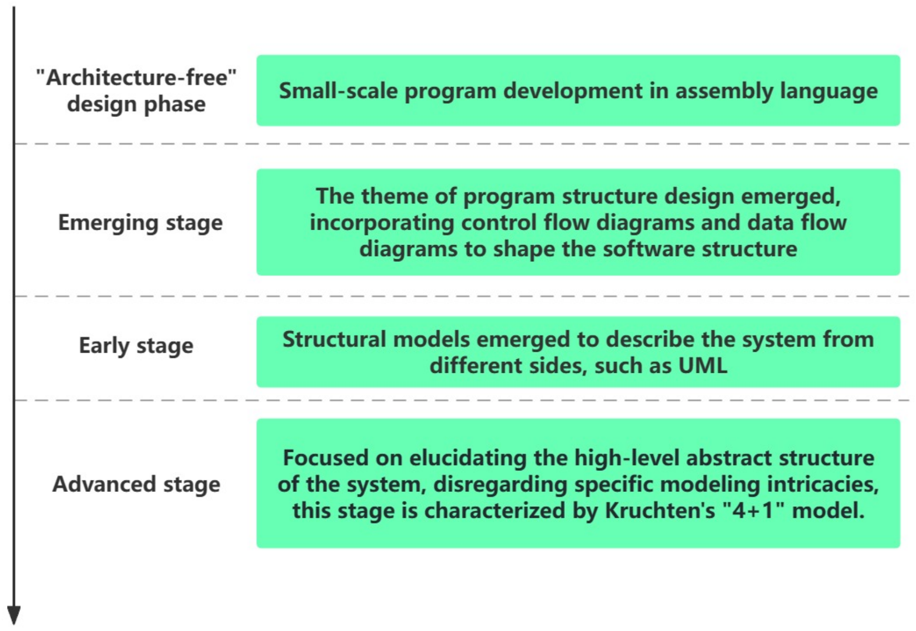

"Architecture-free" design phase
Small-scale program development in assembly language
Emerging stage
The theme of program structure design emerged, incorporating control flow diagrams and data flow diagrams to shape the software structure
Early stage
Structural models emerged to describe the system from different sides, such as UML
Advanced stage
Focused on elucidating the high-level abstract structure of the system, disregarding specific modeling intricacies, this stage is characterized by Kruchten's "4+1" model.

Figure 1: four stages of software architecture

Examining the historical trajectory of software architecture, one can discern four distinct stages, spanning from the initial phase of "unstructured" design to the contemporary paradigm of architecture-based software development, as illustrated in Figure 1.

According to the ISO/IEC/IEEE 42010:2022 description of the software architecture [23], it can be abstracted into the following set:

$$SA = \{system, environment, Architecture, ArchitectureDescription\}$$

1) The term “System” pertains to an entity whose configuration holds significance and can be characterized as ”software-intensive” within the software development process. This designation encompasses any system wherein software plays a substantial role in shaping its design, assembly, deployment, and progression. It encompasses individual applications, conventional systems, subsystems, systems of systems, product lines, product families, entire enterprises, and aggregates of other pertinent entities.
2) The “Environment” encompasses the full spectrum of influences acting upon a system throughout its entire life cycle, which includes its interactions with the surrounding milieu. A system is situated within an environment that may encompass multiple other systems.
3) “Architecture” encompasses the fundamental constituents of a system and its nexus with the environment. This may encompass the constituent elements or parts of the system, their configuration, interconnections, the governing principles of organization, and the design and evolutionary principles that govern the system across its life cycle.
4) An “Architecture Description” is employed to explicate the substance of the system’s architecture. The ensuing discourse offers an exhaustive scrutiny and analysis of the methodology for delineating architecture.

Overall, software architecture encompasses the intricate process of meticulously planning and defining the comprehensive structure and organization of a software system during its design phase. This endeavor entails abstracting and elucidating the various constituents, modules, components, and their interrelationships within the system, with the ultimate aim of realizing the system’s overarching functionality and performance prerequisites. The primary objective of software architecture is to furnish a meticulously crafted blueprint that delineates the responsibilities and functionalities of each system module, as well as establishes the interfaces and interactions between them. This blueprint serves as an invaluable guide to aid the development team in comprehending and effectively communicating the system’s holistic structure and design intent throughout the design phase. Moreover, software architecture provides a well-defined framework that steers developers in module development, component integration, and system testing during the implementation phase. It is crucial to note that software architecture not only focuses on the static structural aspects of the system but also encompasses the dynamic behavior and evolutionary trajectory of the system. The entire process of software architecture is shown in Figure 2. The pro-

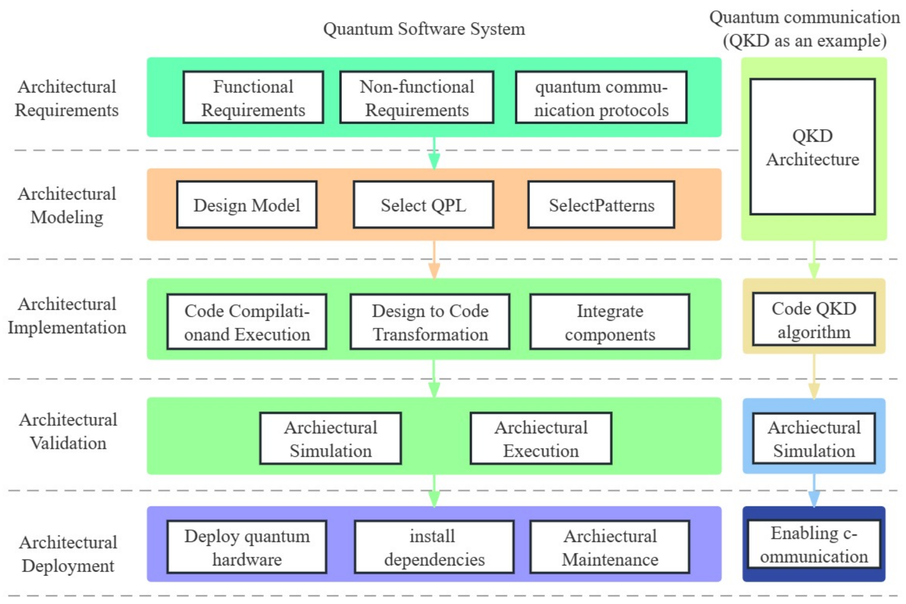


Figure 2: Architectural process and activities

cess comprises five primary activities: architectural requirements, architectural modeling, architectural implementation, architectural validation, and architectural deployment. The architectural requirements activity aims to ascertain the requirements and constraints of quantum software. The architectural modeling activity aims to employ appropriate modeling techniques and tools to describe the architecture of quantum software. The architectural implementation activity aims to transform architectural design into executable code. The architectural validation activity aims to verify the correctness, reliability, and performance of quantum software. The architectural deployment activity aims to deploy quantum software into the target environment. The process also encompasses several supporting activities such as architectural review, architectural evolution, and architectural management. These activities and supporting elements collectively constitute the quantum software architecture process. The design decisions pertaining to software architecture invariably entail a meticulous consideration of system scalability, maintainability, reusability, performance, security, and other pertinent factors.

All software development methods address the transition from requirements to implementation, and quantum software development is no exception. In quantum computing systems, the software architecture represents the blueprint for developing quantum hardware-operated software systems and applications [24]. However, most quantum software projects nowadays focus on generating quantum source code and neglect quantum software design, which is often prone to errors. Software architecture is one of the critical steps in software development, which can guide all stages of software development and has an important impact on the maintainability, scalability, and reusability of software. Software architecture for quantum computing systems enables software engineers to create a model that serves as the basis for system implementation. The model can drive developers to understand requirements and design software to ensure that the software system meets business needs and technical requirements while organizing code and modules and defining interfaces and interactions between modules.

3. Quantum Software Architecture Practice

In the discourse surrounding QSA Practice, its pivotal role and positioning within the software lifecycle are paramount. Given its status as an emerging technology, the inherent potentials and challenges posed by quantum computing demand a comprehensive reassessment of all aspects related to software development and design. This segment aims to scrutinize the placement and significance of QSA within the software lifecycle, examining its influence on the progression of software and the methodologies involved in its development.

3.1. The position of QSA in the software life cycle

Figure 3 illustrates the interconnectedness between the software life cycle and the software architecture process within the broader context of software development. The software development process encompasses various stages, spanning from conceptualization to implementation. These stages include problem definition, requirements analysis, outline design, detailed design, testing, and operation and maintenance. The software architecture is established after the requirements analysis phase and before the software design phase. The following analysis explores the interrelationships among these stages, elucidating the purpose of each phase.

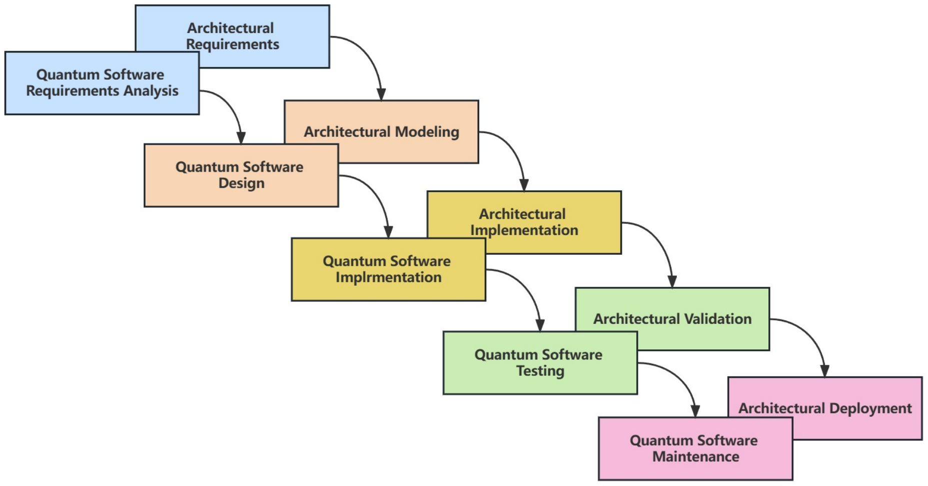

Architectural Requirements
Quantum Software Requirements Analysis
Architectural Modeling
Quantum Software Design
Architectural Implementation
Quantum Software Implrmentation
Architectural Validation
Quantum Software Testing
Architectural Deployment
Quantum Software Maintenance

Figure 3: Architestural Process and quantum software life cycle

1) Requirements analysis phase: This phase primarily focuses on determining the system's functions based on the specified requirements. During this phase, the designer conducts a comprehensive investigation of the target object and its environment. This process involves gathering essential information about the target object and extracting valuable insights. Abstract thinking and logical reasoning are employed to produce software specifications.
2) Software architecture establishment stage: In this stage, the designer analyzes the entire system primarily from a structural perspective. The objective is to select appropriate components, define their interactions, and identify any constraints imposed on them. The ultimate goal is to formulate a system framework that aligns with the user's requirements, thereby establishing the foundation for subsequent design activities.
3) Design phase: The main objective of this phase is to modularize the system and determine the detailed interfaces between the components, algorithms, and data types. These design decisions support the framework developed in the architecture phase and provide the basis for implementation.
4) Implementation phase: In this phase, the algorithms and data types designed during the previous design phase are translated into a programming language. This process ensures that the design, architecture, and require-

ments analysis specifications are met, resulting in the development of a target system that fulfills the design requirements.

The software architecture assumes a pivotal role throughout the entire system development process. It serves as the starting point and foundation for the subsequent design activities. Furthermore, it acts as a guiding principle for system assembly and maintenance.

3.2. The role of QSA in the software life cycle

Software architecture assumes a fundamental and indispensable role throughout the entirety of the system development process, serving as the cornerstone and foundation for design activities. Moreover, it serves as a guiding principle for system assembly and maintenance. The significance of sound software architecture becomes evident across all stages of the software life cycle, and its applicability extends to numerous scenarios within the domain of quantum software engineering.

1) Design and Implementation of Quantum Algorithms: The software architecture for quantum computing is utilized to design and implement quantum algorithms. It provides a structured approach to describe and organize the various components of an algorithm, such as quantum gate operations, quantum registers, and quantum measurements. By employing a well-designed software architecture, the complexity of algorithms can be better managed and organized, enhancing their readability, maintainability, and scalability [25].
2) Facilitating the scalability and reusability of software systems: Quantum computing software architectures play a pivotal role in fostering the scalability and reusability of software systems. By partitioning the software system into discrete modules and components, each entrusted with distinct functions and responsibilities, the software system can be rendered more amenable to scalability and reusability. Quantum computing software architecture encompasses a repertoire of design principles and patterns, such as loose coupling and high cohesion, which empower developers to proficiently devise and execute scalable and reusable quantum computing software systems.
3) Quantum Computing Task Scheduling and Optimization: The software architecture for quantum computing is utilized for task scheduling and optimization. In quantum computing, task scheduling involves mapping

different parts of an algorithm onto the hardware resources of a quantum computer to achieve efficient computations. The software architecture provides a framework and algorithms for task scheduling, aiding in the optimization of task allocation and scheduling to maximize the utilization of the quantum computer's performance.

4. Research Priorities in Quantum Software Architecture

The domain of QSA is marked by a constellation of distinctive challenges. To begin with, both the hardware and software underpinning quantum computers are in their formative stages, necessitating the cultivation of innovative software tools and methodologies. Furthermore, the operational paradigm of quantum computers deviates from that of classical counterparts, necessitating a reevaluation of established approaches to software design and development. Additionally, the elevated error rates intrinsic to quantum computing demand the adoption of specialized error correction coding techniques. As quantum computers continue to burgeon in scale and intricacy, it becomes imperative to conceive novel architectures and algorithms capable of meeting these exigencies. Moreover, the assurance of security and privacy in quantum communication stands forth as a pivotal challenge. The ensuing discourse probes into three critical dimensions: architectural design patterns, architecture modeling languages, and quantum communication. Each of these components constitutes a crucial realm of investigation, proffering potential resolutions to the challenges at hand.

4.1. Quantum Software Modeling Language

As quantum computing technologies have evolved and become more widely used, it has been realized that a formal and precise description method is needed to help understand and design quantum computing systems.

The evolution of classical computing offers valuable lessons. Software modeling, an essential facet, aids in comprehending systems, capturing requirements, architecting software, risk identification, and verifying correctness. UML stands out among traditional modeling languages, widely accepted for its comprehensive nature [26].

Another prominent language, Business Process Modeling Notation (BPMN), serves as a graphical depiction of business processes across diverse sectors like finance, healthcare, and business. BPMN's standardized graphical elements—activities, events, gateways, and process connections—enable clear representation. Its usage empowers companies to comprehend, streamline, and enhance business processes, ultimately boosting efficiency [27].

In the realm of quantum software modeling, the emergence of Q-UML showcased as an extension of the established UML, and Quantum4BPMN brings unique attributes, as highlighted in Table 1. These quantum-centric languages offer avenues for precise depiction of quantum computing systems, addressing their unique characteristics and complexities.

While UML and BPMN remain prominent, numerous other software modeling languages, such as Petri nets [28], temporal logic [29], and SysML [30], cater to diverse domains and possess distinct advantages. Optimal selection of the modeling language aligning with the project's requirements enhances developers' understanding, system depiction, and team collaboration.

Table 1: Q-UML vs. Quantum4BPMN Comparison

Aspect	**Q-UML[31]**	**Quantum4BPMN [32]**
Type	Modeling	Process Modeling
Application	Software Design	Business Process Optimization
Symbols	Class Diagrams, Sequence Diagrams, Use Case Diagrams	Flowcharts, Activity Diagrams, Gateways
Audience	Developers, Architects	Analysts, Designers, Users
Precision	High Detail	Moderate, Focuses on Sequence
Learning	Challenging	Relatively Easy
Generality	Versatile	Business Process Focus
Tools	Abundant Modeling Tools	Specialized BPM Tools
Automation	Requires Development	Direct Support, Integrates with Tools
Focus	Technical	Business
Scenarios	Software Development, System Design	Business Process Management, Optimization, Automation

4.1.1. Q-UML

UML is a widely used modeling language that can be used to describe and design various components and interactions in software systems. components, system structures, and processes. By using UML, developers can better understand the overall structure of a software system and better communicate and collaborate during the design and development process. However, the traditional UML modeling language mainly models classical computing systems, while quantum computing systems have their unique features and requirements, such as the superposition of quantum states and entanglement. These features require a new modeling language to support the description and analysis [33].

In this context, Q-UML, an extension of the classical computer software modeling language UML, was introduced at the Quantum Software Engineering Workshop of the ACM/IEEE International Conference on Software Engineering (ICSE) in 2020. It integrates the characteristics of classical UML with those of quantum computing, offering a formal methodology to describe quantum computing systems [34]. Using Q-UML can help developers better understand and design quantum computing systems, thus improving development efficiency and software quality. Regarding the modeling notation, Q-UML has proposed possible extensions to class and sequence diagrams.

Table 2: Summary View of Q-UML (CD = Class Diagram, UD = Use case Diagram, SD = State Diagram, AD = Activity Diagram, SD = Sequence Diagram, DD = Deployment Diagram)

Ref.	CD	UD	SD	AD	SD	DD
[31]	✓	✓	✓	✓	✓	
[34]	✓				✓	
[35]	✓	✓		✓	✓	✓
[36]	✓					
[37]	✓					
[38]					✓	

Table 2 presents an overview of the attributes associated with Q-UML as a chosen modeling notation by researchers in recent years. Numerous researchers have adhered to the established conventions of modeling in classical software engineering, including the continued utilization of class diagrams.

4.1.2. Quantum4BPMN

Within the realm of quantum computing, Quantum4BPMN, as an extension of BPMN in the quantum domain [32], can be employed as a suitable quantum modeling language to effectively depict and scrutinize business processes within quantum computing systems. By utilizing the BPMN notation, quantum modeling languages can aptly represent various concepts associated with quantum algorithms [39], quantum circuits [40], and quantum communications. As an integral component of the quantum modeling language, Quantum4BPMN offers a user-friendly and comprehensible approach to delineating and evaluating business processes within quantum computing systems. By leveraging its capabilities, developers can enhance their comprehension and proficiency in quantum algorithm design, thereby fostering advancements and practical implementations in the domain of quantum computing.

4.2. Architecture Design Patterns

Software architecture design patterns are a set of generic solutions for solving common software design problems. This notion of pattern and pattern language has its origin in [41]. A pattern is a structured document containing an abstract description of a proven solution to a recurring problem. At present, software architects have not yet devised specific architecture patterns tailored for quantum software systems. Such patterns are essential for effectively addressing quantum-related concerns while specifying the architecture of quantum software systems [42]. For instance, the layered architecture pattern entails dividing a system into distinct layers, each with defined functions and responsibilities. However, there exists a compatibility issue between the software architectural design patterns of classical and quantum computing systems. This arises from the fundamental disparities in their computational models and programming paradigms: classical computation operates based on Boolean logic principles and manipulates bits, whereas quantum computation operates on quantum bits, leveraging principles from quantum mechanics. This implies that software architectural design patterns tailored for classical computing may not directly apply to quantum computing. On the other hand, quantum software development lacks standardization. Currently, there are several QPLs and development frameworks available, each with its own set of design patterns and best practices. This makes it challenging to devise a unified software architectural design pattern applicable across different quantum software systems. In order to address these compatibility issues, researchers are delving into novel software architectural design patterns specifically tailored for quantum computation. These design patterns take into account the unique characteristics of quantum computation, such as superposition, entanglement, and interference, providing a framework for designing efficient, scalable, and reliable quantum software systems. Table 3 provides a comparative analysis of four commonly employed architectural design patterns for quantum software system architecture from various perspectives.

Table 3: Comparison of Architectural Design Patterns

Criteria	**Layered Architecture**	**Pipe and Filter**	**Prototype Pattern**	**Two-Qubit Gate Pattern**
Advantages				
Modularity	High	Medium	Low	Medium
Concurrency	Low	High	Low	High
Object Creation Time	-	-	High	-
Dynamic Object Addition/Removal	Medium	High	Medium	Medium
Testability	High	Medium	Medium	Medium
Quantum Parallelism	-	-	-	High
Supports Complex Quantum Operations	-	-	-	High
Disadvantages				
Performance Overhead	Medium	Low	Low	High
Complexity	Medium	Low	Low	Medium
Communication Overhead	-	Medium	-	-
Requires Hardware Support	-	-	-	Yes
Applicability				
Clear Software Layering	Yes	-	-	-
Processing Large Data Streams	-	Yes	-	-
Creating Many Similar Objects	-	-	Yes	-
Need for Fast Parallel Processing	-	Yes	-	-
Utilizing Quantum Parallelism for Speedup	-	-	-	Yes

4.2.1. Layered Pattern

Among the existing studies, the Layered pattern has the most research, which has found extensive application in numerous software development scenarios, particularly in the design and development processes of large-scale, complex systems. The layered architecture was first proposed in [43], which supports framework development in a hierarchical and systematic manner that allows challenges to be solved independently at each level. Each layer has assigned responsibilities, and interfaces are defined between interacting layers, with lower layers providing services and processing commands from higher layers. This approach allows quantum engineers to focus on individual challenges while also seeing how a process fits into the overall design. The framework needs to address issues such as defective hardware, error management, and classical processing for quantum computer design. However, this leads to some degree of performance loss due to the need for communication between the different layers. Fu et al. [44] also adopted a layered model design approach in designing the Heterogeneous Quantum Computer Architecture with quantum error correction and detailed the stack structure of the quantum computing system and quantum compiler.

4.2.2. Pipe and Filter Pattern

The Pipeline and Filter architectural pattern is a widely embraced paradigm in parallel programming, designed to streamline the efficient handling of data. This pattern finds extensive application in scenarios that involve the processing of data streams, such as log processing, image manipulation, and text analysis. Its primary function lies in breaking down intricate systems into smaller, more manageable components, thereby enhancing their development, testing, and maintenance. This decomposition occurs through a series of sequential processing stages termed filters, wherein each filter conducts specific data operations and then passes the processed data to the subsequent filter in the pipeline, as illustrated in Figure 4. The essence of this pattern revolves around fostering principles of modularity, reusability, and adaptability, thus laying a robust foundation for software design.

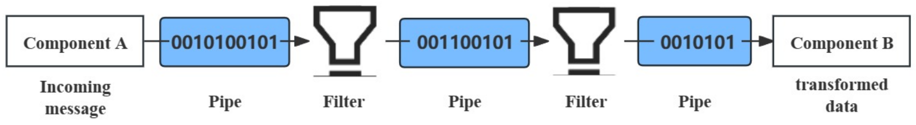


Figure 4: Pipe and filter pattern

Initially introduced by Buschmann et al. [45], the Pipeline and Filter pattern has undergone successive refinements by various researchers. The modern interpretation of this pattern demonstrates enhanced characteristics in terms of security, maintainability, and extensibility. Notably favored by scholars involved in quantum computing research [46], it is particularly esteemed due to its inherent simplicity, operational efficiency, and ease of debugging.

However, it is important to acknowledge that the Pipeline and Filter pattern, like any architectural approach, is not without its limitations. One notable drawback lies in the potential performance overhead introduced by the necessity for inter-filter communication and data transfer, especially when dealing with large volumes of data.

4.2.3. Prototype Design Pattern

The main idea of the prototype design pattern is to use a prototype object as a template and then duplicate this object to create new objects. For example, if we need to create multiple identical quantum states in our program, then we can use a prototype object as a template and then copy this object to create new quantum states. This will avoid re-creating a new object each time, thus improving the efficiency and performance of the program [47].

The Prototype design pattern is a flexible and efficient approach to object creation, particularly suitable for scenarios where the creation process is intricate or there is a need to circumvent the overhead of constructors. This makes it very flexible and scalable in quantum software. In addition, using the prototype design pattern reduces the overhead of object creation, thus improving the efficiency and performance of the program. Figure 5 shows the structure of the Prototype pattern.

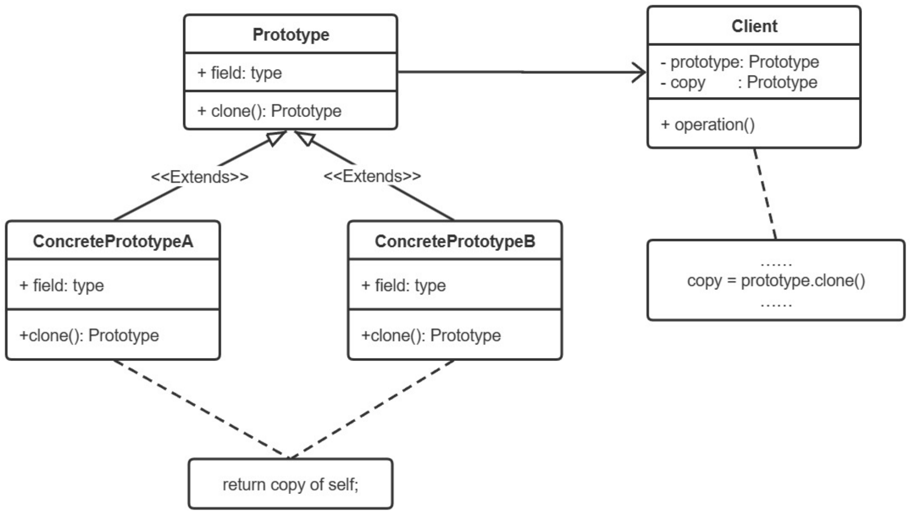


Figure 5: Prototype design pattern

However, the prototype design pattern also has some disadvantages. First, it may consume a lot of memory because of the need to copy objects. Second, because of the complex relationships that may exist between objects, these relationships need to be taken into account when copying objects, which may increase the complexity of the code.

4.2.4. Two-qubit Gate Pattern

Two-qubit gate pattern is applicable to numerous quantum computing tasks, particularly in the construction and operation of quantum circuits. The basic idea is to take two quantum bits as input and perform a specific gate operation on them to obtain the output.

Advantages of this mode of door operation include:

- Scalability: The two-qubit gate pattern can be extended to an arbitrary number of gate operations between quantum bits to build more complex quantum circuits and algorithms.
- Controllability: The two-qubit gate pattern can be gated between any two quantum bits, allowing flexible control of the interactions and exchanges between quantum bits.
- Reusability: The two-qubit gate pattern is a generic gate operation pattern that can be reused in multiple quantum circuits and algorithms to save development time and resources [48].

However, there are some challenges with the two-qubit gate pattern:

- Resource consumption: The two-qubit gate pattern usually requires a large number of hardware resources and energy, which is a challenge for the practical implementation of quantum computing systems [49].
- Errors and noise: Due to the volatility and noise of quantum bits, the two-qubit gate pattern may be affected by errors and noise, which leads to unreliability and inaccuracy of quantum computing results.

As a consequence, the two-qubit gate pattern is a useful mode of gate operation, but some challenges and limitations need to be overcome to achieve reliable and efficient quantum computing.

4.3. Quantum Communication

One current focal point in quantum science involves advancing communication efficiency [50]. This entails the transmission of more information using fewer carriers, leveraging the quantum state of physical entities such as photons, atoms, or ions as carriers for information encoding. These quantum states are transmitted through quantum channels to convey information [51]. Additionally, quantum computing demonstrates the capability to encrypt transmitted information. In classical computing, established cryptographic methodologies like blockchain technology [52, 53], Software-Defined Networking (SDN) [54], and select machine learning algorithms [55] can be integrated with quantum computing, yielding heightened efficiency and bolstered security in algorithmic pursuits. The communication process adheres to fundamental principles of quantum mechanics such as Heisenberg's uncertainty principle and the theory of measurement collapse. Quantum communication primarily encompasses Quantum Key Distribution, Quantum Dense Coding, and Quantum Teleportation.

4.3.1. Quantum Key Distribution

Quantum Key Distribution (QKD) technology is the most important and mainstream technology in the current research and application of quantum confidential communication. By providing information-theoretically secure key distribution, it opens a new chapter in the use of quantum methods for secure transmission [56].

In conventional encryption methods, both communicating entities employ a shared key for both encryption and decryption operations. However, this approach introduces the inherent vulnerability of the key being compromised by malicious adversaries. In contrast, QKD leverages quantum bits as cryptographic keys. These qubits are generated by the sender via quantum randomization techniques and subsequently transmitted to the receiver through quantum communication channels. Notably, the transmission process incorporates mechanisms to detect any unauthorized measurement attempts on the quantum bits, as per the principles of quantum mechanics. Illicit measurements can destroy the key, safeguarding its confidentiality. Figure 6 visually illustrates this key destruction mechanism, highlighting the interplay between quantum mechanics and the security of QKD. In the initialization phase, Alice encodes single photons according to specific schemes and transmits them to Bob, who prepares measurement devices. Upon receiving the quantum states, Bob randomly selects a basis (vertical, horizontal,

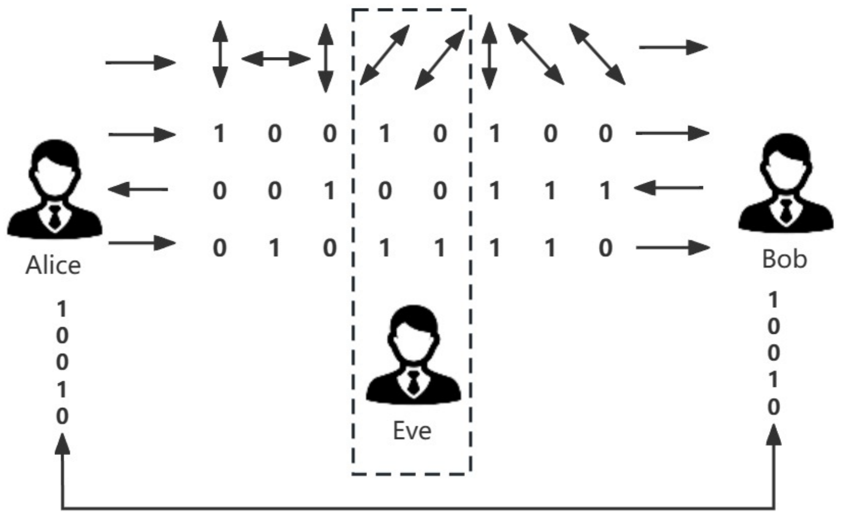


Figure 6: Quantum key distribution(Alice and Bob represent the two sides of the communication, Eve is the eavesdropper)

diagonal) for measurement and records the corresponding outcomes. Subsequently, Alice and Bob publicly disclose a portion of their measurement results for discussion, aiming to detect any eavesdropping attempts. Due to the non-clonability of quantum states, illicit measurements lead to state collapse, thereby being detected during the public discussion phase. In the event of anomalies, Alice and Bob abandon the current key and terminate communication. Otherwise, they process the measurement outcomes to derive a shared secure key. Nonetheless, adversaries may attempt to intercept quantum states multiple times during transmission, potentially introducing disruptions even if eavesdropping is detected.

Table 4: Literature Summary of QKD

Ref.	Transmission Distance	Key Rate	Protocol	Average Attenuation
[57]	1200km	1000HZ	downlink protocol, BB84	-
[58]	600km	1bps	TF-QKD	-
[59]	833km	-	TF-QKD	0.1419 dB/km
[60]	202.81km	6.214bps	CV-QKD	-
[61]	511km	3.45bps	TF-QKD	0.158dB/km
[62]	10km	115.8Mb/s	BB84	-
[63]	1002km	47.06Kbps	SNS-TF-QKD	0.157dB/km

QKD technology is in the initial stage of radicalization. there are two main ways to implement QKD, one is discrete variable quantum key distribution (DV-QKD) and continuous-variable quantum key distribution (CV-QKD). In the first approach, single-photon detectors are usually used to decode. In the second approach, the quantum states are described in the optical domain, and their special states are continuous and have infinite dimensions [64]. The two have a good complementary relationship and each has its own application focus: in coding, CV-QKD cannot achieve continuous coding in reality, which brings experimental complexity and discretization errors; while in detection, the coherent detectors of CV-QKD are better than the SPDs used in DV-QKD [65]. In light of recent advances in QKD, an increasing number of researchers have embraced the utilization of the Twin-Field Quantum Key Distribution (TF-QKD) protocol within their experimental endeavors. This strategic choice is driven by the imperative to surmount the constraints pertaining to transmission rates and distances, which have been inherent challenges in the domain of QKD. Table 4 presents a comprehensive compilation of empirical findings achieved by esteemed scholars in the realm of QKD over the course of recent years.

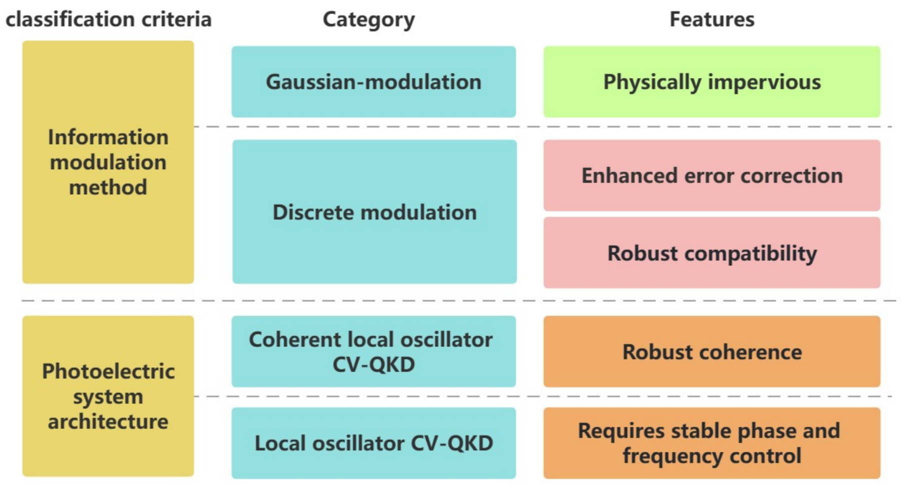

classification criteria
Category
Features
Information modulation method
Gaussian-modulation
Physically impervious
Discrete modulation
Enhanced error correction
Robust compatibility
Photoelectric system architecture
Coherent local oscillator CV-QKD
Robust coherence
Local oscillator CV-QKD
Requires stable phase and frequency control

Figure 7: Classification and characteristics of CV-QKD

CV-QKD is mainly divided into Gaussian modulation type protocols and discrete modulation type protocols according to the different information modulation methods. From the perspective of optoelectronic system architecture, there are existing CV-QKD systems with accompanying local oscillation CV-QKD schemes and local oscillation CV-QKD schemes. Figure 7 presents the classification and distinctive characteristics of the four types of CV-QKD.

4.3.2. Quantum Dense Coding

Quantum Dense Coding (QDC), a technique based on quantum entanglement, was proposed by Bennett et al. [66] in 1992 as one of the most intriguing applications embodying the properties of quantum entanglement.

QDC relies on a specific form of quantum entanglement known as Bell states. Bell states represent a maximally entangled state of a two-qubit system, exhibiting the following characteristics:

- Maximal Entanglement: Bell states achieve the highest possible degree of entanglement between two qubits. This implies that upon measuring one qubit, the state of the other qubit instantaneously collapses to a definite outcome.

- Non-Locality: Bell states exhibit non-local correlations, meaning that a measurement of one qubit instantaneously influences the state of the other qubit, even if they are separated by considerable distances.

A concrete example of a Bell state is

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \tag{4}$$

where $|00\rangle$ and $|11\rangle$ denote the basis states of a two-qubit system. This state holds particular significance as it forms the foundation for realizing QDC.

QDC uses classical bits to represent information and exploits the nonlocal correlation property of quantum entangled states to send one quantum bit and transmit two bits of classical information using a quantum channel, which has attracted the research interest of a wide range of scholars. Bennett's quantum dense coding scheme considers one sender and one receiver, and in 1998, Bose et al. [67] extended the dense coding scheme to a multiparty information transmission process. In 2010, Situ et al. [68] proposed a simultaneous dense coding scheme for one sender and multiple receivers. Zhang et al. [69] analyzed the performance of the simultaneous dense coding protocol in the case of decoherence of quantum states. In 2014, Situ et al. [70] proposed another controlled simultaneous dense coding scheme based on GHZ states. Currently, numerous scholars are ardently engaged in the refinement and advancement of QDC methodologies. Table 5 serves to encapsulate a comprehensive compendium of extant experimental endeavors undertaken in the realm of QDC theory.

Table 5: Literature Summary of QDC

Ref.	Coding Scheme	Quantum Entanglement Scheme
[68]	Simultaneous Dense Coding	Bell State, GHZ State, W State
[70]	Simultaneous Dense Coding	GHZ State
[71]	Superdense Coding	Bell State
[72]	Device-independent Quantum Dense Coding	Bell State
[73]	Distributed Quantum Dense Coding	GHZ State
[74]	Distributed Quantum Dense Coding	GHZ State
[75]	Superdense Coding	Bell State
[76]	Private Dense Coding	Bell State
[77]	Time-Bin Encoding	Bell State
[78]	Time-Bin Encoding	Bell State

4.3.3. Quantum Teleportation

Quantum Teleportation (QT) was first proposed by Bennett et al. [79]. QT is a communication technique based on quantum entanglement that allows the transmission of a quantum state from one place to another without replicating or transmitting the actual quantum state in the intermediate process. The keys to achieving quantum invisible state transfer are quantum entanglement and quantum measurement. Technology is also mentioned in the famous science fiction novel *Three Bodies*. However, in reality, because QT requires classical channels to transmit information, it is theoretically impossible for the information to be transmitted faster than the speed of light.

Figure 8 reveals the basic principle of QT: Alice performs a measurement on an unknown quantum state, represented by a particle intended for transmission, using one of the particles from an entangled pair known as an EPR pair, employing the Bell basis. According to fundamental principles of quantum mechanics, it is established that the quantum information encoded within the unknown quantum state is transferred to the other particle of the entangled EPR pair. Subsequently, the measurement outcome is conveyed to Bob through a classical communication channel. Bob, upon receiving this information, gains confidence in the state-carrying particle (the other particle

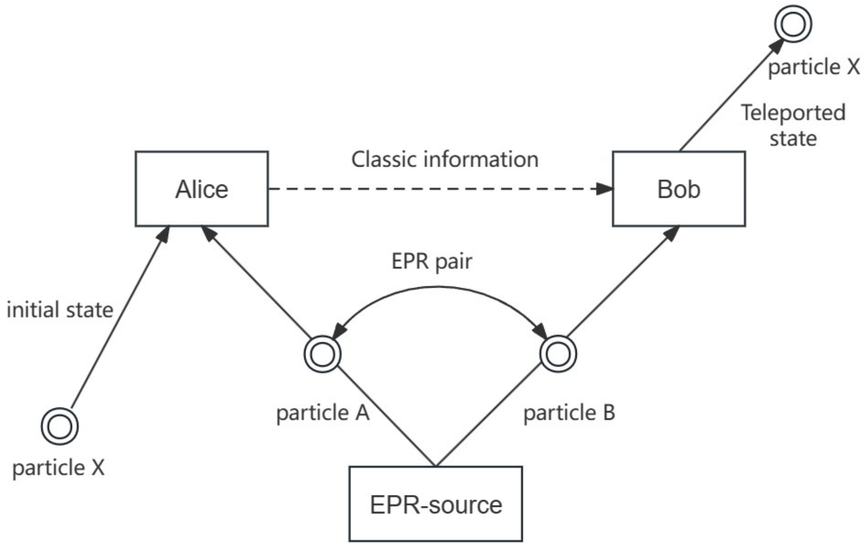


Figure 8: The principle of QT

of the EPR pair) and proceeds to reproduce the original quantum state. The measurement result is then communicated back to Bob through the classical channel, ensuring that the carrier particle can acquire a state similar to the one originally transmitted. Through this process, the faithful reproduction of the quantum state is achieved.

Table 6: Literature Summary of QT

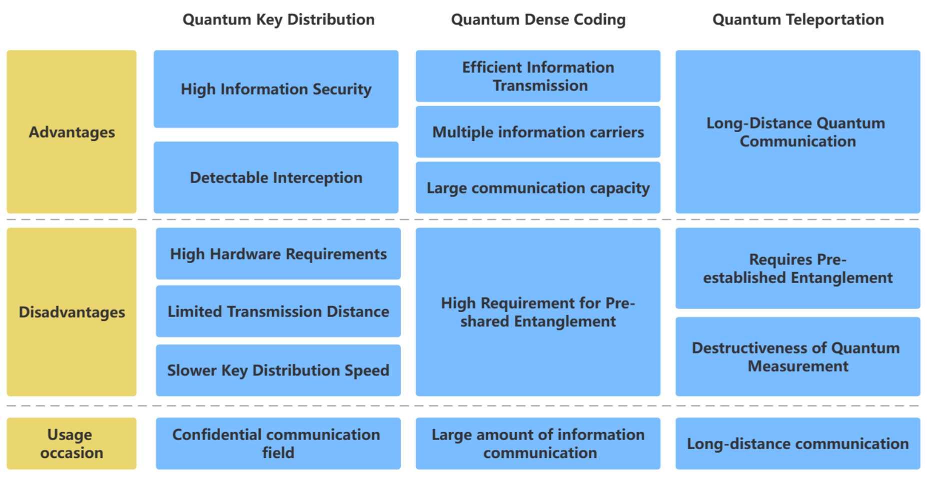
Ref.	Implementation Platform	Entanglement Type	Measurement	Communication Channel
[80]	Nanobeams	EPR	BSM	Optical transmission channel
[81]	Single photon	qutrit-qutrit	HD-BSM	Optical transmission channel
[82]	CQI-MP	three-dimensional	HD-BSM	Quantum communication channel
[83]	MRR	Bell operator	MZI	Photons transmitted within chip
[84]	Quantum circuit	EPR	BSM	Classical communication channel

* BSM = Bell-State Measurement, HD-BSM = High-Dimensional Bell-State Measurement, CQI-MP = Collective quantum interference of multiple particles, MRR = Microresonator, MZI = Mach-Zehnder Interferometer

QT relies on quantum entanglement and quantum measurements without transmitting the actual quantum state. Although quantum stealth transfer still requires the support of quantum channels, it can effectively avoid the impact of channel noise and interference on transmission quality and thus has a wide range of applications. Table 6 summarizes the QT experiments in recent years. Nowadays, quantum teleportation experiments have diversified their platforms, including optical systems, MRRs on silicon chips [83], and linear-optic multiqubit circuits. Novel techniques, such as nonlinear optical sources and HD-BSM, have been introduced to improve the generation efficiency and transmission quality of photon pairs [81]. Research has explored various types of entangled pairs, such as EPR pairs and three-dimensional entangled states. Additionally, catalytic teleportation has been studied, involving an additional quantum system shared between Alice and Bob, aiming to enhance the teleportation process without affecting the catalyst state [85].

4.3.4. Performance Comparison and Applicability Scenarios

The preceding elucidation of QKD, QDC, and QT necessitates a comparative evaluation of their operational efficacy and domain-specific suitability. The merits, demerits, and practical scenarios of the three quantum communication methods are delineated in Figure 9.



	Quantum Key Distribution	Quantum Dense Coding	Quantum Teleportation
Advantages	High Information Security; Detectable Interception	Efficient Information Transmission; Multiple information carriers; Large communication capacity	Long-Distance Quantum Communication
Disadvantages	High Hardware Requirements; Limited Transmission Distance; Slower Key Distribution Speed	High Requirement for Pre-shared Entanglement	Requires Pre-established Entanglement; Destructiveness of Quantum Measurement
Usage occasion	Confidential communication field	Large amount of information communication	Long-distance communication

Figure 9: Characteristics of Communication Methods

QKD emerges as an exemplar of cryptographic security protocols. Grounded in the principles of quantum mechanics, QKD guarantees information confidentiality via inherent properties of quantum states, rendering intercepted transmissions fundamentally unobtainable. However, practical implementation demands sophisticated quantum hardware and is limited by transmission distances constrained by present technological capabilities.

QDC showcases superior efficiency in information transmission. By leveraging entanglement and encoding classical bits into quantum states, it enables higher transmission rates than classical methods. This efficiency proves beneficial in bandwidth-limited scenarios, yet it necessitates a pre-shared entangled state, posing practical challenges in its application.

QT excels in quantum state transfer across vast distances. Its ability to teleport quantum information over extensive separations without physically traversing the intervening space holds promise for secure long-distance communication. Nonetheless, its reliance on entanglement establishment and the requirement for classical communication channels for teleportation completion present practical constraints.

5. Conclusions

To solve the problem of quantum programmers ignoring the overall quantum software design, QSA has emerged as a new type of software architecture. Many researchers have made contributions to this field [86]. At the same time, we migrate the research methods and tools from the classical domain to the quantum realm, we should also pay attention to the characteristics of the quantum field.

Implications and Future Work: At present, quantum modeling languages are existing models that are simplified versions of classical modeling methods and do not explicitly cover quantum properties, including superposition, interference, and entanglement, which will make it difficult to exploit the full features of quantum computing in the designed software. The design models for QSA are poorly developed, and there is also a lack of decision methods for selecting design models. Researchers in [21] identified and prioritized challenges associated with QSA execution processes and devised a comprehensive decision framework to address these issues, opening new avenues for subsequent studies.

In the domain of quantum communication, the establishment of quantum networks presents a significant challenge. There is an urgent requirement for communication protocols tailored to quantum networks. Similarly, end-users play a crucial role in the development process. Currently, traditional software development methods lack practical experience in quantum software development, necessitating researchers' exploration in this domain. Khan et al. proposed attempts to integrate agile practices into quantum software development in [87].

In conclusion, while serving as a vital tool for quantum software development, the software architecture for quantum computing requires further research and widespread adoption. As part of future work, we aim to apply the insights gleaned from this research to enhance our processes and undertake additional studies to evaluate these improvements.

References

[1] Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61(2):117, 1998.

[2] Sukhpal Singh Gill, Adarsh Kumar, Harvinder Singh, Manmeet Singh, Kamalpreet Kaur, Muhammad Usman, and Rajkumar Buyya. Quantum computing: A taxonomy, systematic review and future directions. *Software: Practice and Experience*, 52(1):66–114, 2022.

[3] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Dec 2002.

[4] Akshay Ajagekar and Fengqi You. New frontiers of quantum computing in chemical engineering. *Korean Journal of Chemical Engineering*, 39(4):811–820, 2022.

[5] Thomas E O'Brien, Michael Streif, Nicholas C Rubin, Raffaele Santagati, Yuan Su, William J Huggins, Joshua J Goings, Nikolaj Moll, Elica Kyoseva, Matthias Degroote, et al. Efficient quantum computation of molecular forces and other energy gradients. *Physical Review Research*, 4(4):043210, 2022.

[6] Raihan Ur Rasool, Hafiz Farooq Ahmad, Wajid Rafique, Adnan Qayyum, Junaid Qadir, and Zahid Anwar. Quantum computing for healthcare: A review. *Future Internet*, 15(3):94, 2023.

[7] Md Sajid Anis, Héctor Abraham, R Agarwal AduOffei, Gabriele Agliardi, Merav Aharoni, Ismail Yunus Akhalwaya, Gadi Aleksandrowicz, Thomas Alexander, Matthew Amy, Sashwat Anagolum, et al. Qiskit: An open-source framework for quantum computing. *Qiskit/qiskit*, 2021.

[8] Jie Zhang, Jiawei Lu, Xuan Yan, Xiaolong Xu, Lianyong Qi, and Wanchun Dou. Quantified edge server placement with quantum encoding in internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):9370–9379, 2021.

[9] Shaohua Wan, Lianyong Qi, Xiaolong Xu, Chao Tong, and Zonghua Gu. Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications*, 25:743–755, 2020.

[10] Zhijie Shen, Bowen Liu, Xiaolong Xu, Lianyong Qi, Fei Dai, and Wanchun Dou. Iiot mobile business data placement strategy based on bayesian optimization algorithm. In *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pages 186–193. IEEE, 2022.

[11] Xiaolong Xu, Qingxiang Liu, Yun Luo, Kai Peng, Xuyun Zhang, Shunmei Meng, and Lianyong Qi. A computation offloading method over big data for iot-enabled cloud-edge computing. *Future Generation Computer Systems*, 95:522–533, 2019.

[12] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q# enabling scalable quantum computing and development with a high-level dsl. In *Proceedings of the real world domain specific languages workshop 2018*, pages 1–10, 2018.

[13] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, F Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, et al. Qiskit: An open-source framework for quantum computing. *Accessed on: Mar*, 16, 2019.

[14] Brian Fitzgerald and Klaas-Jan Stol. Continuous software engineering and beyond: trends and challenges. In *Proceedings of the 1st International Workshop on rapid continuous software engineering*, pages 1–9, 2014.

[15] Nathalie P De Leon, Kohei M Itoh, Dohun Kim, Karan K Mehta, Tracy E Northup, Hanhee Paik, BS Palmer, Nitin Samarth, Sorawis Sangtawesin, and David W Steuerman. Materials challenges and opportunities for quantum computing hardware. *Science*, 372(6539):eabb2823, 2021.

[16] Tore Dyba, Barbara A Kitchenham, and Magne Jorgensen. Evidence-based software engineering for practitioners. *IEEE software*, 22(1):58–65, 2005.

[17] John Clark and Susan Stepney. Quantum software engineering. In *Workshop on Grand Challenges for Computing Research, e-Science Institute, Edinburgh*, 2002.

[18] Susan Stepney, Samuel L Braunstein, John A Clark, Andy Tyrrell, Andrew Adamatzky, Robert E Smith, Tom Addis, Colin Johnson, Jonathan Timmis, Peter Welch, et al. Journeys in non-classical computation i: A grand challenge for computing research. *International Journal of Parallel, Emergent and Distributed Systems*, 20(1):5–19, 2005.

[19] Luis S Barbosa. Software engineering for'quantum advantage'. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 427–429, 2020.

[20] Aakash Ahmad, Muhammad Waseem, Peng Liang, Mahdi Fehmideh, Arif Ali Khan, David Georg Reichelt, and Tommi Mikkonen. Engineering software systems for quantum computing as a service: A mapping study. *arXiv preprint arXiv:2303.14713*, 2023.

[21] Muhammad Azeem Akbar, Arif Ali Khan, and Saima Rafi. A systematic decision-making framework for tackling quantum software engineering challenges. *Automated Software Engineering*, 30(2):22, 2023.

[22] Mario Piattini, Guido Peterssen, and Ricardo Pérez-Castillo. Quantum computing: A new software engineering golden age. *ACM SIGSOFT Software Engineering Notes*, 45(3):12–14, 2021.

[23] Ademir AC Júnior, Sanjay Misra, and Michel S Soares. A systematic mapping study on software architectures description based on iso/iec/ieee 42010: 2011. In *Computational Science and Its*

Applications–ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, July 1–4, 2019, Proceedings, Part V 19, pages 17–30. Springer, 2019.

[24] Arif Ali Khan, Aakash Ahmad, Muhammad Waseem, Peng Liang, Mahdi Fahmideh, Tommi Mikkonen, and Pekka Abrahamsson. Software architecture for quantum computing systems—a systematic review. *Journal of Systems and Software*, 201:111682, 2023.

[25] Frank Leymann and Johanna Barzen. Hybrid quantum applications need two orchestrations in superposition: a software architecture perspective. *arXiv preprint arXiv:2103.04320*, 2021.

[26] Lvar Jacobson and James Rumbaugh Grady Booch. The unified modeling language reference manual. 2021.

[27] Mark von Rosing, Stephen White, Fred Cummins, and Henk de Man. Business process model and notation-bpmn., 2015.

[28] Wolfgang Reisig. *Petri nets: an introduction*, volume 4. Springer Science & Business Media, 2012.

[29] Nicholas Rescher and Alasdair Urquhart. *Temporal logic*, volume 3. Springer Science & Business Media, 2012.

[30] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[31] Carlos A Pérez-Delgado. A quantum software modeling language. In *Quantum Software Engineering*, pages 103–119. Springer, 2022.

[32] Benjamin Weder, Uwe Breitenbücher, Frank Leymann, and Karoline Wild. Integrating quantum computing into workflow modeling and execution. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 279–291. IEEE, 2020.

[33] Ricardo Pérez-Castillo, Luis Jiménez-Navajas, and Mario Piattini. Modelling quantum circuits with uml. In *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, pages 7–12. IEEE, 2021.

[34] Carlos A Pérez-Delgado and Hector G Perez-Gonzalez. Towards a quantum software modeling language. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 442–444, 2020.

[35] Ricardo Pérez-Castillo and Mario Piattini. Design of classical-quantum systems with uml. *Computing*, 104(11):2375–2403, 2022.

[36] Alexander J McCaskey, Eugene F Dumitrescu, Dmitry Liakh, Mengsu Chen, Wu-chun Feng, and Travis S Humble. A language and hardware independent approach to quantum–classical computing. *SoftwareX*, 7:245–254, 2018.

[37] Hongbo Lan, Chengrui Zhang, and Hongbin Li. An open design methodology for automotive electrical/electronic system based on quantum platform. *Advances in Engineering Software*, 39(6):526–534, 2008.

[38] Vincenzo Pisano. Plugin-based workflow integration for qhana. B.S. thesis, 2022.

[39] Benjamin Weder, Johanna Barzen, Martin Beisel, and Frank Leymann. Analysis and rewrite of quantum workflows: Improving the execution of hybrid quantum algorithms. In *CLOSER*, pages 38–50, 2022.

[40] Benjamin Weder, Johanna Barzen, Frank Leymann, and Marie Salm. Automated quantum hardware selection for quantum workflows. *Electronics*, 10(8):984, 2021.

[41] Christopher Alexander et al. Alexander c., ishikawa s., silverstein m.,a pattern language, 1977.

[42] Frank Leymann. Towards a pattern language for quantum algorithms. In *Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings 1*, pages 218–230. Springer, 2019.

[43] Krysta M Svore, Alfred V Aho, Andrew W Cross, Isaac Chuang, and Igor L Markov. A layered software architecture for quantum computing design tools. *Computer*, 39(1):74–83, 2006.

[44] Xiang Fu, Leon Riesebos, Lingling Lao, Carmen G Almudever, Fabio Sebastiano, Richard Versluis, Edoardo Charbon, and Koen Bertels. A heterogeneous quantum computer architecture. In *Proceedings of the ACM International Conference on Computing Frontiers*, pages 323–330, 2016.

[45] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerland, and Michael Stal. Layers. *Pattern-Oriented Software Architecture: a system of patterns*, 1:31–32, 1996.

[46] Qiong Li, Dan Le, and Ming Rao. A design and implementation of multi-thread quantum key distribution post-processing software. In *2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pages 272–275. IEEE, 2012.

[47] S Ihnatchenko, V Kudin, and S Homeniuk. Application of prototype design pattern in parallel implementation of finite element analysis systems. *Publishing House "Baltija Publishing"*, 2021.

[48] Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Physical Review A*, 69(3):032315, 2004.

[49] Seyed Shakib Vedaie, Eduardo J Páez, Nhung H Nguyen, Norbert M Linke, and Barry C Sanders. Bespoke pulse design for robust rapid two-qubit gates with trapped ions. *arXiv e-prints*, pages arXiv–2212, 2022.

[50] Nicolas Gisin and Rob Thew. Quantum communication. *Nature photonics*, 1(3):165–171, 2007.

[51] Daniele Cozzolino, Beatrice Da Lio, Davide Bacco, and Leif Katsuo Oxenløwe. High-dimensional quantum communication: benefits, progress, and future challenges. *Advanced Quantum Technologies*, 2(12):1900038, 2019.

[52] Zixin Wang, Bin Cao, Chenxi Liu, Congfang Xu, and Lei Zhang. Blockchain-based fog radio access networks: Architecture, key technologies, and challenges. *Digital Communications and Networks*, 8(5):720–726, 2022.

[53] Zheng Yan, Qinghua Zheng, Yulei Wu, Yaliang Zhao, and Mohammed Atiquzzaman. Guest editorial: Blockchain-enabled technologies for cyber-physical systems and big data applications, 2022.

[54] Wenjuan Li, Yu Wang, Zhiping Jin, Keping Yu, Jin Li, and Yang Xiang. Challenge-based collaborative intrusion detection in software-defined networking: an evaluation. *Digital Communications and Networks*, 7(2):257–263, 2021.

[55] Jorge Gallego-Madrid, Ramon Sanchez-Iborra, Pedro M Ruiz, and Antonio F Skarmeta. Machine learning-based zero-touch network and service management: A survey. *Digital Communications and Networks*, 8(2):105–123, 2022.

[56] Charles H Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *arXiv preprint arXiv:2003.06557*, 2020.

[57] Sheng-Kai Liao, Wen-Qi Cai, Wei-Yue Liu, Liang Zhang, Yang Li, Ji-Gang Ren, Juan Yin, Qi Shen, Yuan Cao, Zheng-Ping Li, et al. Satellite-to-ground quantum key distribution. *Nature*, 549(7670):43–47, 2017.

[58] Mirko Pittaluga, Mariella Minder, Marco Lucamarini, Mirko Sanzaro, Robert I Woodward, Ming-Jun Li, Zhiliang Yuan, and Andrew J Shields. 600-km repeater-like quantum communications with dual-band stabilization. *Nature Photonics*, 15(7):530–535, 2021.

[59] Shuang Wang, Zhen-Qiang Yin, De-Yong He, Wei Chen, Rui-Qiang Wang, Peng Ye, Yao Zhou, Guan-Jie Fan-Yuan, Fang-Xiang Wang, Wei Chen, et al. Twin-field quantum key distribution over 830-km fibre. *Nature photonics*, 16(2):154–161, 2022.

[60] Yichen Zhang, Ziyang Chen, Stefano Pirandola, Xiangyu Wang, Chao Zhou, Binjie Chu, Yijia Zhao, Bingjie Xu, Song Yu, and Hong Guo. Long-distance continuous-variable quantum key distribution over 202.81 km of fiber. *Physical review letters*, 125(1):010502, 2020.

[61] Jiu-Peng Chen, Chi Zhang, Yang Liu, Cong Jiang, Wei-Jun Zhang, Zhi-Yong Han, Shi-Zhao Ma, Xiao-Long Hu, Yu-Huai Li, Hui Liu, et al. Twin-field quantum key distribution over a 511 km optical fibre linking two distant metropolitan areas. *Nature Photonics*, 15(8):570–575, 2021.

[62] Wei Li, Likang Zhang, Hao Tan, Yichen Lu, Sheng-Kai Liao, Jia Huang, Hao Li, Zhen Wang, Hao-Kun Mao, Bingze Yan, et al. High-rate quantum key distribution exceeding 110 mb s–1. *Nature Photonics*, 17(5):416–421, 2023.

[63] Yang Liu, Wei-Jun Zhang, Cong Jiang, Jiu-Peng Chen, Chi Zhang, Wen-Xin Pan, Di Ma, Hao Dong, Jia-Min Xiong, Cheng-Jun Zhang, et al. Experimental twin-field quantum key distribution over 1000 km fiber distance. *Physical Review Letters*, 130(21):210801, 2023.

[64] Christian Weedbrook, Stefano Pirandola, Raúl García-Patrón, Nicolas J Cerf, Timothy C Ralph, Jeffrey H Shapiro, and Seth Lloyd. Gaussian quantum information. *Reviews of Modern Physics*, 84(2):621, 2012.

[65] Liyun Hu, M Al-Amri, Zeyang Liao, and MS Zubairy. Continuous-variable quantum key distribution with non-gaussian operations. *Physical Review A*, 102(1):012608, 2020.

[66] Charles H Bennett and Stephen J Wiesner. Communication via one- and two-particle operators on einstein-podolsky-rosen states. *Physical review letters*, 69(20):2881, 1992.

[67] Sougato Bose, Vlatko Vedral, and Peter L Knight. Multiparticle generalization of entanglement swapping. *Physical Review A*, 57(2):822, 1998.

[68] Haozhen Situ and Daowen Qiu. Simultaneous dense coding. *Journal of Physics A: Mathematical and Theoretical*, 43(5):055301, 2010.

[69] Cai Zhang, Haozhen Situ, Qin Li, and Guang Ping He. Efficient simultaneous dense coding and teleportation with two-photon four-qubit cluster states. *International Journal of Quantum Information*, 14(05):1650023, 2016.

[70] Haozhen Situ. Controlled simultaneous teleportation and dense coding. *International Journal of Theoretical Physics*, 53:1003–1009, 2014.

[71] Brian P Williams, Ronald J Sadlier, and Travis S Humble. Superdense coding over optical fiber links with complete bell-state measurements. *Physical review letters*, 118(5):050501, 2017.

[72] Nayana Das and Goutam Paul. Device-independent quantum secure direct communication with user authentication. *arXiv preprint arXiv:2304.03201*, 2023.

[73] Dagmar Bruß, G Mauro D'Ariano, Maciej Lewenstein, Chiara Macchiavello, A Sen, Ujjwal Sen, et al. Distributed quantum dense coding. *Physical review letters*, 93(21):210501, 2004.

[74] Tamoghna Das, R Prabhu, Aditi Sen, Ujjwal Sen, et al. Distributed quantum dense coding with two receivers in noisy environments. *Physical Review A*, 92(5):052330, 2015.

[75] T Schaetz, Murray D Barrett, Dietrich Leibfried, J Chiaverini, Joseph Britton, Wayne M Itano, John D Jost, Christopher Langer, and David J Wineland. Quantum dense coding with atomic qubits. *Physical review letters*, 93(4):040505, 2004.

[76] Jiawei Wu, Gui-Lu Long, and Masahito Hayashi. Quantum secure direct communication with private dense coding using a general preshared quantum state. *Physical Review Applied*, 17(6):064011, 2022.

[77] Philipp Kurpiers, Marek Pechal, Baptiste Royer, Paul Magnard, Theo Walter, Johannes Heinsoo, Yves Salathé, Abdulkadir Akin, Simon Storz, J-C Besse, et al. Quantum communication with time-bin encoded microwave photons. *Physical Review Applied*, 12(4):044067, 2019.

[78] Ilaria Vagniluca, Beatrice Da Lio, Davide Rusca, Daniele Cozzolino, Yunhong Ding, Hugo Zbinden, Alessandro Zavatta, Leif K Oxenløwe, and Davide Bacco. Efficient time-bin encoding for practical high-dimensional quantum key distribution. *Physical Review Applied*, 14(1):014051, 2020.

[79] Charles H Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Physical review letters*, 70(13):1895, 1993.

[80] Niccolò Fiaschi, Bas Hensen, Andreas Wallucks, Rodrigo Benevides, Jie Li, Thiago P Mayer Alegre, and Simon Gröblacher. Optomechanical quantum teleportation. *Nature Photonics*, 15(11):817–821, 2021.

[81] Xiao-Min Hu, Chao Zhang, Bi-Heng Liu, Yu Cai, Xiang-Jun Ye, Yu Guo, Wen-Bo Xing, Cen-Xiao Huang, Yun-Feng Huang, Chuan-Feng Li, et al. Experimental high-dimensional quantum teleportation. *Physical Review Letters*, 125(23):230501, 2020.

[82] Yi-Han Luo, Han-Sen Zhong, Manuel Erhard, Xi-Lin Wang, Li-Chao Peng, Mario Krenn, Xiao Jiang, Li Li, Nai-Le Liu, Chao-Yang Lu, et al. Quantum teleportation in high dimensions. *Physical review letters*, 123(7):070505, 2019.

[83] Daniel Llewellyn, Yunhong Ding, Imad I Faruque, Stefano Paesani, Davide Bacco, Raffaele Santagati, Yan-Jun Qian, Yan Li, Yun-Feng Xiao, Marcus Huber, et al. Chip-to-chip quantum teleportation and multi-photon entanglement in silicon. *Nature Physics*, 16(2):148–153, 2020.

[84] Angela Sara Cacciapuoti, Marcello Caleffi, Rodney Van Meter, and Lajos Hanzo. When entanglement meets classical communications: Quantum teleportation for the quantum internet. *IEEE Transactions on Communications*, 68(6):3808–3833, 2020.

[85] Patryk Lipka-Bartosik and Paul Skrzypczyk. Catalytic quantum teleportation. *Physical Review Letters*, 127(8):080502, 2021.

[86] Arif Ali Khan, Pekka Abrahamsson, and Mahmood Niazi. Introduction to the special issue on managing software processes using soft computing techniques, 2022.

[87] Arif Ali Khan, Muhammad Azeem Akbar, Aakash Ahmad, Mahdi Fahmideh, Mohammad Shameem, Valtteri Lahtinen, Muhammad Waseem, and Tommi Mikkonen. Agile practices for quantum software development: Practitioners' perspectives. In *2023 IEEE International Conference on Quantum Software (QSW)*, pages 9–20. IEEE, 2023.