Modelling Quantum Software: An Annotated Bibliography

Modelado de software cuántico: una bibliografía comentada

Modelagem de software quântico: uma bibliografia anotada

Luis Mariano Bibbo [1,(*)], Alejandro Fernandez [2],
Jose Manuel Suarez [3], Oscar Pastor [4]

Recibido: 10/10/2024 Aceptado: 10/10/2024

Summary. - This annotated bibliography delves into the field of quantum computing, focusing specifically on the resources used throughout the software life cycle. It examines several published works that analyze quantum software modelling in the context of the various phases of the life cycle, from analysis/requirements to testing and maintenance. Each annotation provides an analysis of software engineering resources applicable to quantum software development and their applicability to different phases of the software development process. By synthesizing these diverse perspectives, this bibliography illuminates the evolving landscape of quantum software development and underscores the critical role of modelling in the context of software engineering. The result provides a valuable starting point for researchers and practitioners who wish to deepen the interplay between quantum computing and software engineering, fostering innovation and advances in this evolving field.

Keywords: Quantum Software Engineering, Quantum Computing, Quantum Modelling, Quantum Design.

(*) Corresponding Author.

[1] Researcher. LIFIA, Facultad de Informatica, Universidad Nacional de La Plata, La Plata, Argentina. lmbibbo@lifia.info.unlp.edu.ar, ORCID iD: https://orcid.org/0000-0003-4950-3605

[2] Researcher. LIFIA, Facultad de Informatica, Universidad Nacional de La Plata, CICBA - Provincia de Buenos Aires La Plata, Argentina. casco@lifia.info.unlp.edu.ar, ORCID iD: https://orcid.org/0000-0002-7968-6871

[3] Researcher. LIFIA, Facultad de Informatica, Universidad Nacional de La Plata, La Plata, Argentina. jsuarez@lifia.info.unlp.edu.ar, ORCID iD: https://orcid.org/0009-0001-1115-6225

[4] Researcher. PROS Research Centre, Universitat Politecnica de Valencia, Valencia, Espana. opastor@dsic.upv.es, ORCID iD: https://orcid.org/0000-0002-1320-8471

Memoria Investigaciones en Ingeniería, núm. 27 (2024). pp. 285-301
https://doi.org/10.36561/ING.27.19
ISSN 2301-1092 • ISSN (en línea) 2301-1106 – Universidad de Montevideo, Uruguay

Resumen. - *Esta bibliografía anotada profundiza en el campo de la computación cuántica, centrándose específicamente en los recursos utilizados a lo largo del ciclo de vida del software. Examina varios trabajos publicados que analizan el modelado de software cuántico en el contexto de las diversas fases del ciclo de vida, desde el análisis/requisitos hasta las pruebas y el mantenimiento. Cada anotación proporciona un análisis de los recursos de ingeniería de software aplicables al desarrollo de software cuántico y su aplicabilidad a diferentes fases del proceso de desarrollo de software. Al sintetizar estas diversas perspectivas, esta bibliografía ilumina el panorama cambiante del desarrollo de software cuántico y subraya el papel fundamental del modelado en el contexto de la ingeniería de software. El resultado proporciona un valioso punto de partida para investigadores y profesionales que deseen profundizar en la interacción entre la computación cuántica y la ingeniería de software, fomentando la innovación y los avances en este campo en evolución.*

Palabras clave: *Ingeniería de software cuántico, Computación cuántica, Modelado cuántico, Diseño cuántico.*

Resumo. ***-*** *Esta bibliografia anotada se aprofunda no campo da computação quântica, focando especificamente nos recursos usados ao longo do ciclo de vida do software. Ela examina vários trabalhos publicados que analisam a modelagem quântica de software no contexto das várias fases do ciclo de vida, desde análise/requisitos até testes e manutenção. Cada anotação fornece uma análise dos recursos de engenharia de software aplicáveis ao desenvolvimento de software quântico e sua aplicabilidade a diferentes fases do processo de desenvolvimento de software. Ao sintetizar essas diversas perspectivas, esta bibliografia ilumina o cenário em evolução do desenvolvimento de software quântico e ressalta o papel crítico da modelagem no contexto da engenharia de software. O resultado fornece um ponto de partida valioso para pesquisadores e profissionais que desejam aprofundar a interação entre computação quântica e engenharia de software, promovendo inovação e avanços neste campo em evolução.*

Palavras-chave: *Engenharia de Software Quântico, Computação Quântica, Modelagem Quântica, Design Quântico.*

1. Introduction. - Software engineering, according to Boehm [1], ”is the practical application of scientific knowledge to the design, development of computer programs and the associated documentation required to develop, operate and maintain them”. It is also known as Software Development or Software Production. Bauer [2], treats Software Engineering as the establishment of engineering principles and methods for obtaining costeffective software that is reliable and works on real machines. Then in 1993, the IEEE synthesised it as the application of a systematic, disciplined and quantifiable approach to software development, operation and maintenance.

At its core, software engineering addresses the challenges associated with creating and maintaining complex and reliable software systems. This involves carrying out activities such as requirements specification, architectural design, code implementation, software testing, configuration management, project management and software quality assurance.

One of the main objectives of software engineering is to improve productivity and efficiency in software development, while minimizing errors and ensuring customer satisfaction. This is achieved through the application of software development practices and methodologies, such as Model Driven Development (MDD) [3], [4], Extreme Programming (XP), Agile methodology, among others [5].

1.1. Quantum Computing. - Quantum computing is an innovative field that relies on the principles of quantum mechanics to process information in a radically different way from classical computing [6]. Unlike classical bits, which can be in one of two states (0 or 1) at any time, quantum qubits can exist in multiple states simultaneously [7], thanks to a phenomenon known as quantum superposition. This allows quantum computing to explore a much wider range of possibilities in parallel, making it potentially much more powerful for certain types of problems.

One of the fundamental concepts in quantum computing is quantum entanglement, which allows two or more qubits to be intrinsically related so that the state of one is instantaneously correlated with the state of the other, regardless of the distance between them. This leads to a phenomenon known as quantum teleportation, where information can be transferred between distant qubits without requiring physical transmission of the information through space [8].

Quantum algorithms [9], such as Shor’s famous algorithm for integer factorization and Grover’s algorithm for unstructured search, offer enormous potential for solving problems that would be virtually impossible to tackle with classical computation in a reasonable amount of time. These algorithms take advantage of the unique properties of quantum mechanics to perform computations much more efficiently than their classical counterparts, which could have a significant impact in areas such as cryptography, optimization and molecular simulation.

Despite its potential, quantum computing is still in its early stages of development and faces numerous technical challenges, such as the need to build more robust and stable qubits, develop effective quantum error correction techniques and build scalable hardware architectures. However, recent advances in the field have generated great interest and optimism in the scientific and technological community, and quantum computing is expected to play a transformative role in solving some of the most complex challenges in computing and science in the decades to come [10].

Quantum computing represents an innovative way of processing information by exploiting the principles of quantum mechanics. These computing systems, based on units of information called qubits, operate in a superposition state, allowing them to perform calculations simultaneously in multiple states. *Quantum algorithms* are algorithms designed specifically to be run on quantum computers, taking advantage of the unique properties of quantum mechanics to solve problems more efficiently than their classical counterparts. *Quantum circuits*, on the other hand, are graphical representations of sequences of quantum operations that transform a set of input qubits into a desired result. These circuits are the basis for the implementation of quantum algorithms and are composed of *Quantum gates*, which represent the fundamental logical operations in quantum computation, such as the Hadamard gate, the CNOT gate and the phase gate. Each quantum gate performs a specific transformation on the states of qubits, allowing complex computations and efficient problem solving in a quantum environment.

1.2. Modelling and Quantum Computing. - Quantum computing has emerged as a fascinating field that promises to revolutionize the way we process information, and its potential impact extends to a wide range of fields, from cryptography and artificial intelligence to molecular simulation and process optimization. In this context, modelling plays a crucial role in providing tools and methodologies to represent and understand quantum systems more accurately and effectively. Through modelling, researchers and developers can capture the complex phenomena underlying quantum computation, explore new possibilities for quantum algorithms and circuits, and design more efficient and robust systems. In this paper, we will analyze in depth various initiatives that are exploring modelling techniques applied to Quantum Computing. This review is inspired by the work of kitchenham [11] where we will focus on examining how different modelling approaches are being used to advance the field of quantum computing, identifying the challenges and opportunities they face and exploring their potential to drive the next generation of quantum technologies.

As quantum computing advances, the need to develop higher-level tools and programming languages to facilitate the development of quantum or mixed systems becomes evident. Currently, most of the tools available for quantum computing are expressed at a low level, similar to assembly language in classical computing. This means that programmers must have a deep understanding of the underlying architecture and the specific characteristics of qubits to write effective code, which can be a significant obstacle for those who are not experts in quantum physics.

To address this challenge, it is essential to develop higher-level software engineering resources that hide the underlying complexity and allow developers to focus on business logic and problem solving [12]. This includes creating more intuitive programming languages and software abstractions that simplify interaction with quantum systems. In addition, there is a need to develop libraries and frameworks that provide common functions and development tools for tasks such as simulation, debugging and optimization of quantum algorithms.

By favoring the development of quantum or mixed systems, where parts of the system are quantum and others are traditional [10], these software engineering resources can pave the way for the wider adoption of quantum computing in a variety of applications and industry sectors. By enabling a broader set of developers to harness the power of quantum computing without requiring a deep understanding of the underlying physics, these tools have the potential to accelerate innovation and bring quantum computing out of the lab and into the real world.

Building quantum software benefits greatly from the use of abstract models to guide the development process. These models provide a simplified but accurate representation of quantum systems, allowing developers to capture the essence of their operation and behavior without having to worry about implementation details at the code level. By focusing on creating these abstract models, developers can concentrate on properly understanding and specifying system requirements, facilitating clearer and more effective communication between team members and ensuring a shared understanding of the project.

The advantage of using abstract models in building quantum software lies in their ability to simplify the inherent complexity of quantum systems. By representing the essential aspects of the system in a clear and concise manner, abstract models allow developers to approach complex problems more systematically and effectively. Furthermore, by separating the specification of the system from its implementation, abstract models make it easier to adapt the software as requirements or technologies evolve, ensuring that the system is flexible and can be easily maintained over time.

In summary, the use of abstract models in building quantum software offers a number of significant benefits, including better understanding and specification of system requirements, clearer communication between team members, and greater flexibility and adaptability in software development [12]. By focusing on the creation and use of these abstract models, organizations can develop quantum software more efficiently and effectively, driving the adoption and advancement of quantum computing in a variety of applications and fields of study. Understanding ”how” abstract models are used for building ”what” type of quantum software is one of the goals of the work that we present here.

2. Papers Analyzed. - The following compilation features a curated selection of various software engineering resources within the domain of Quantum Software modelling. Each entry delves into the complexities of quantum

software modelling, elucidating the intricate interplay between quantum principles and established software engineering practices.

First, we will write a brief summary of each work, and then we will review the collection to identify a) which software engineering resources they contribute and b) at which stage of the software development lifecycle they are applied.

1. (2019) [13] *Towards a Pattern Language for Quantum Algorithms:*

The paper presents a set of Abstract Patterns for Quantum programming. These are de list of patterns proposed:

- Initialisation (State Preparation)
- Uniform Superposition
- Creating Entanglement
- Function Table
- Oracle (Black Box)
- Uncompute (Unentangling, Copy-Uncompute)
- Phase Shift
- Amplitude Amplification
- Speedup via Verifying
- Quantum-Classic Split

Patterns describe abstract solutions, independent of any concrete implementations. They intend to grow the proposed pattern language and make it available in the pattern repository PatternPedia. In parallel, concrete implementations of the patterns in quantum languages like QASM are considered, and these implementations will be linked to the corresponding patterns.

2. (2020) [14] *Towards a Quantum Software Modeling Language*, (2022) [15] *Design of classical-quantum systems with UML* also in [16] *Chapter 6: A quantum software modeling language:*

In these works, the authors introduce the concept of Quantum software modelling by expressing that Quantum Computing is based on the counter-intuitive principles of quantum mechanics [7], such as superposition and entanglement, among others. Thus, lessons learned from existing and classical design techniques cannot simply be applied [9], but new quantum foundations and features have to be devised. In this sense, quantum software is currently still coded manually and ad hoc. For example, although there are some well-known algorithms that are defined in the literature using mathematical formalism, these algorithms have been hand-coded in multiple ways for specific quantum programming languages without any prior design or modelling. They then introduce the concept of - Hybrid quantum information systems: The adoption of quantum information systems will not be a complete replacement of classical information systems. Both technologies will operate in parallel in what is known as ”hybrid information systems”. This is because it does not make sense to use quantum software to solve simple and simplistic problems, as these are already adequately performed by classical computers, and at lower cost. Thus, classical software will in the future act as a driver orchestrating requests to quantum software. Classical quantum information systems must therefore be analyzed and designed together, which in itself poses another major challenge.

The main contribution of this work is to provide a quantum UML profile that covers the main modelling aspects of the analysis and design of hybrid information systems. The quantum UML profile also covers structural and behavioral aspects. In particular, it addresses use cases, class, sequence, activity and deployment diagrams. As a result, both quantum and classical elements, as well as the relationships between them, can be modelled together in an integrated design.

It has a special section discussing Quantum Software Design, where different approaches are mentioned.

3. (2020) [17] *Software engineering for 'quantum advantage':*

This paper argues for the need for a proper quantum software engineering discipline benefiting from precise foundations and calculi, capable of supporting algorithm development and analysis.

They identify the following main issues around which any roadmap for a Software Engineering discipline should take into account: i) how (quantum) systems are *modelled*, ii) how models are *c*omposed (*architectures*), and finally, iii) how *properties* of their behaviors are anticipated, expressed and verified.

- *Models:* Different effects in computational models, for example partiality, non-determinism, probabilism, etc, are formally captured by monads (a structure that in functional programming combines program fragments "functions" and wraps their return values in a type with additional computation). In particular, this may be a way to bring quantum and cyber-physical programming together. Actually, recent developments on the semantics of (classical) hybrid components introduced a new monad capturing continuous evolution in a topological setting which caters for stability aspects.
- *Architectures:* A first challenge in this direction will aims at exploring what an algebra of mixed classical and quantum systems could be. In a sense the mathematical structure underlying 'categorical quantum mechanics already possesses the basic ingredients for a calculus of quantum processes: composition (via tensor), measurements of entangled states (allowed by the compact closed structure), feedback (through dagger) and probabilistic branching (via biproducts), and last but not least, a formal diagrammatic notation.
- *Properties:* Three research challenges emerge: i) the characterization of a contract-based development discipline for assertional reasoning about quantum programs; ii) the proposal of a dynamic logic incorporating (classical) noise at the expression level to reason about probabilistically controlled fault tolerance in quantum programs; and, finally, iii) the identification of a semantic bridge between 'categorical quantum mechanics', which may be seen as a type theoretic form of quantum logic, and dynamic logics supporting assertional reasoning.

Their starting point, similar to classical computation, quantum algorithmics will significantly benefit from a mathematically based approach. This approach should be capable of conceptualizing and predicting behavior, as well as providing a rich, formal framework for specifying, developing, and verifying quantum algorithms. These foundational aspects emerge from the confluence of various mathematical disciplines and insights gleaned from software engineering practice.

4. (2020) [18] *Selection of quantum computing architecture using a decision tree approach:*

The motivation of this paper is to discuss various architecture examples from different industries and then to create a standard decision tree to choose the right software architecture depending on the nature of the quantum project.

The different types of projects are Quantum annealing, Quantum simulation and Universal quantum computing (Gate-based quantum algorithms). They developed a questionnaire to identify the type of project. The questions are simple but quite trivial.

5. (2020) [19] *Integrating quantum computing into workflow modelling and execution:*

They introduce a modelling extension for imperative workflow languages to enable the integration of quantum computations and ease the orchestration of classical applications and quantum circuits. They validate the practical feasibility of our approach by applying our proposed extension to BPMN and introduce Quantum4BPMN.

6. (2020) [20] *Modeling Quantum programs: Challenges, initial results, and research directions:*

The paper defines the concepts of high-level abstraction (platform independent) modeling language development. Proposes a quantum modeling approach at multiple levels of abstraction, class diagramming, flows and state machine. Presents an example of quantum entanglement modeling. It also proposes some lines of action and open issues involving Quantum Software Modeling Notations and Methodologies, Quantum Constraint Specification, Quantum Model-based Testing, Quantum Program Generation among others.

7. (2021) [21] *Towards Model-Driven Quantum Software Engineering:*

This work explores the use of Model Driven Engineering (MDE) on quantum technologies by means of an intermediate abstraction layer (with metamodels, transformations and automatic code generation) that allows the independence of current platforms (hardware + languages) and their interoperability. The solution is described in a very general and abstract way, proposing modelling tools pending development, which were later developed in a subsequent work by the same author [22].

8. (2021) [23] *Quantum Software Models: The Density Matrix for Classical and Quantum Software Systems Design:*

This paper claims that there should be just a single unified and rigorous design procedure for both classical and quantum software systems.

The main contribution of this paper is the unified linear algebra design approach to classical and quantum software systems, starting from their representation as design Density Matrices.

9. (2022) [22] *A Model-Driven Framework for Composition-Based Quantum Circuit Design:*

In this pre-print they propose a composition-oriented modelling language for creating quantum circuits together with a modelling framework which (i) allows for a flexible and convenient definition and application of composite operations including iterative patterns, and (ii) provides automated code generation. Besides that, the proposed approach also comes with a separation between the quantum circuit syntax and the definitions of the quantum operations which allows to build and use customized libraries.

The proposed approach comes with the separation of the quantum operation definitions, from the quantum circuit syntax.

- The meta-model for the quantum circuit design.
- A description of the quantum library which comprises the bespoke definitions of quantum operations.
- Information on certain implemented quantum operations.
- An extension for classical problem-specific inputs for operation definitions.
- Finally, the authors show how quantum circuits can be represented using the proposed framework with a simple example.

10. (2022) [24] *Software Architecture for Quantum Computing Systems – A Systematic Review:*

The objective of this review is to complement Software Engineering based studies and specifically focus on identification, classification, and synthesis of the published research on the role that software architecture plays in developing quantum computing systems. It also introduces the connector as a concept (to link components). This can be useful if we relate it to Quantum-Classical hybrid computing. This work has many good references and covers all stages of quantum software development and quantum architectures.

Although it is not a specific work on modeling, we include this paper because it is a very complete review and has a particular section that talks about modeling. It also has a section that discusses hybrid software architecture modeling (traditional and quantum programming) and software architecture design patterns.

It is important to mention that in modeling topics, the author Khan et al. refers to the same works that we analyze, which are: [18], [13] [14]. We identified in this work some definitions and concepts related to modeling. *Architectural process and activities:* An architecture design endeavour for the quantum software requires an architecting process to incorporate a number of architecting activities. Existing architectural process can be leveraged to support five architecting activities for quantum software namely (i) architectural requirements, (ii) architectural modelling, (iii) architectural implementation, (iv) architectural validation, and (v) architectural deployment.

Architectural modelling notations: Modelling notations to specify quantum software architectures primarily rely on box and arrow notations (having component diagrams) and graph-based models (having state graph) to represent the structures and behaviour of quantum software under design. Unlike conventional software architectures that mostly exploit UML notations (often considered as a defacto approach for software design), there is much less evidence on UML-based modelling quantum software architectures.

It appears that there is a need for architectural description languages and UML profiles that can be helpful to leverage existing tools, frameworks, and architectural knowledge to empower the role of designers and architects to model, develop, and evolve quantum software based on re-usability and (semi-) automation. *Architecture design patterns:* Layered and pipe and filter patterns are identified as the most recurring quantum software architecture patterns. However, these are generic or classical patterns that can be used to design any software system. To this end, further research efforts are required to explore and propose new patterns to particularly focus on quantum computing attributes (e.g., superposition and quantum entanglement) and facilitate the architecture of quantum software systems.

As said the paper is a good starting point to find references to other primary works that focus on different topics related to Quantum Software design and Quantum Architectures.

11. (2023) [25] *Model-Driven Optimization for Quantum Program Synthesis with MOMoT:*

This paper presents how MDO approaches, in particular MOMoT framework, can be applied to the problem of automated quantum program synthesis. They show how MOMoT can interact with quantum SDKs in order to provide the quantum-specific functionalities. Compared to tailor-made solutions for quantum program synthesis, this enables to use the features of existing MDO (Model Driven Optimization) frameworks.

12. (2023) [26] *A Graph-Based Approach for Modelling Quantum Circuits:*

This article proposes a unified metamodel for modelling quantum circuits, together with five strategies for its use and some examples of its application. The article also provides a set of constraints (OCL) for using the identified strategies, a set of procedures for transforming the models between the strategies, and an analysis of the suitability of each strategy for performing common tasks in a model-driven quantum software development environment.

The metamodel extends the well-known metamodel of graphs (direct-acyclic graphs) to combine all the approaches into a single modelling language.

This makes it possible to support different modelling strategies and the transformations that can be applied between them based on the Node concept. In short, the Node provides the developer with the necessary flexibility to decide whether she/he wants to model the quantum circuit using only the gates that operate on a qubit, or to group in a Node all the gates that model a controlled gate, or in the most extreme case, all the gates that appear in a particular vertical section of the circuit.

They also provide a deep study of the metamodel by (i) providing OCL constraints that check whether models are valid, (ii) providing OCL query functions to identify the modelling strategy employed by a model, (iii) outlining model-to-model transformations between the strategies, (iv) outlining a model transformation to reduce the number of modelling elements required to model a quantum circuit, and (v) providing an OCL query function to identify whether

said transformation has been applied to a model.

13. (2023) [27] *A Taxonomic View of the Fundamental Concepts of Quantum Computing–A Software Engineering Perspective:*

In the paper, the authors present a kind of taxonomical view of the fundamental concepts of quantum computing and the derived concepts that integrate the emerging discipline of quantum software engineering. The objective of this review only intends to detect the fundamental concepts of quantum computing and quantum software as an starting point to address the study of this discipline. The paper introduces the concepts of quantum computing and builds an interesting body of Knowledge (BOK).

It then discusses different related works with a special section on Teaching Quantum Computing.

The results of the quasi-systematic review are expressed in a set of informal taxonomies and relationships between concepts. However, as the authors indicate, the taxonomy is a useful starting point to see what is being done in software engineering and quantum computing.

14. (2023) [28] *Formalization of Quantum Intermediate Representations for Code Safety:*

This work proposes the formal definition of a generic reusable compilation tool (qIRs, intermediate representations) based on LLVM and applicable on different frontend languages and back-end hardware platforms. It formally defines syntax and semantics and an intermediate code conversion.

15. (2023) [29] *ScaffML: A Quantum Behavioral Interface Specification Language for Scaffold:*

This paper proposes the use of a behavioral interface specification language (BISL) called ScaffML for the quantum programming language Scaffold to specify behavior and interface of programs, modules, prepostconditions and assertions.

16. (2024) [30] *SimuQ: A Framework for Programming Quantum Hamiltonian Simulation with Analog Compilation:*

Development and implementation of a domain-specific language (SimuQ) for cross-platform quantum Hamiltonian simulation and compilation. Design and implement a Hamiltonian modelling language, an abstract instruction set and a compiler for quantum simulation with an intermediate representation and compilation phase. They apply ad-hoc optimization strategies for each specific platform based on the analysis of these Hamiltonians, which makes it possible to bring the simulated performance closer to the real performance on that platform. SimuQ generates executable pulse schedules for real devices to simulate the evolution of desired quantum systems, which is demonstrated on superconducting (IBM), neutral-atom (QuEra), and trapped-ion (IonQ) quantum devices.

3. Discussion. -

3.1. Quantum design efforts. - Within this section, we embark on a classification of the papers based on the design strategies they employ in the realm of quantum software modelling. By categorizing the papers according to their chosen design strategies, we aim to provide readers with a comprehensive overview of the diverse methodologies employed in the field, enabling them to glean insights and draw parallels across different research endeavors. This classification not only facilitates a deeper understanding of the landscape of quantum software modelling but also lays the groundwork for future exploration and innovation in this rapidly evolving domain.

- Math Oriented: [26] It represents the quantum circuit in a graph and through a metamodel transforms the graph into other analogous representations that may be more efficient.
  In [23] propose to use Density Matrix to Map Classical and Quantum Computing
- Mapping Techniques: Bandic et al. [31] propose a mapping technique based on a structured design for space exploration strategy.

- Workflow Oriented: Weder et al. [19] propose QuantMe as a generic extension for imperative workflow modelling languages.
  Ali and Yue [32] present guidelines for developing quantum software modelling languages and provide a conceptual model of quantum programs as well as an example of modelling based on state machines.
- Optimization, Transformation and Mappings: The work [28] formalizes the functionality of the Quantum Intermediate Representation while respecting the non-cloning theorem and avoid calls to released qubits and arrays.
  The paper [29] presents ScaffML that provides an easy-to-use specification language for quantum programmers, supporting static analysis, run-time checking, and formal verification of Scaffold programs.
- Conceptual Models: Dave Wecker, Krysta M. Svore [33] LIQUi: A Software Design Architecture and Domain-Specific Language for Quantum Computing [17] proposes to deepen the study of three lines of quantum software engineering: Models, Architectures and Properties. [27] Introduces the main concepts of quantum computing and classifies them. Presents a useful taxonomy to introduce the subject.
  Perez Delgado and Perez-Gonzalez in [14] extend UML class diagram to model quantum characteristics in a class design. Then Perez Castillo et. all in [15] Propose a QUML profile (UML extension) for modelling quantum elements together with classical elements.
  The paper [22] is the result of previous papers [21] and [25] by the same author, which work with model-driven methods to produce quantum software. The papers propose modeling languages and transformers that allow the construction of quantum-specific functionalities. The work [20] propose on model-based engineering of quantum programs and assert that they must be platform independent.

3.2. Modelling and Lifecycle. - We will now analyze the works compiled in the previous sections and place them in the different stages identified by the work of Benjamin Weder, Johanna Barzen, Frank Leymann, and Daniel Vietz in chapter 4 of the book *Quantum Software Engineering. Software Engineering* by Manuel A. Serrano, Ricardo Perez-Castillo, and Mario Piattini. [34].

1. (2019) [13] *Towards a Pattern Language for Quantum Algorithms*:
This paper presents a pattern language that mainly addresses the *Modelling* stage of quantum applications. The work is very interesting in terms of teaching and explaining the concepts. It can also be the basis for a broader pattern language covering aspects of Error Mitigation and deployment. It opens up a wide spectrum of areas to be expressed in Patterns and Pattern Language format.

2. (2020) [14] *Towards a Quantum Software Modeling Language*, (2022) [15] *Design of classical-quantum systems with UML* also in [16] *Chapter 6; A quantum software modeling language* Perez-Delgado's work [14] presented in chapter 6 of the book [16]: All these works contribute mainly to the stages of *"Requirements Analysis"* and *"Architecture & Design"*. Its major contribution is to propose a profile of UML which is the most widely used modelling language in traditional software development. The profile allows developers to build hybrid Quantum-Classical designs from different classical UML diagrams. For example, they extended not only static diagrams (Use Cases, Class Diagrams) but also dynamic diagrams such as (Activity and Sequence Diagrams).

They can also be considered as a contribution to the *"Quantum-Classical Splitting"* stage, as the link between classical and quantum circuit designs can be worked out. And they also support the *"Deployment"* stage by extending the UML deployment diagrams. On quantum circuit programming, they help to write what one already knows. They do not provide other perspectives on quantum programming. Other design alternatives may be required to model specific issues in quantum circuits.

A positive aspect is that developers can import the UML diagrams and continue with the design of quantum aspects in it. Many of the known tools for UML could be used to support working with the diagrams (visualization, printing, etc.). Among the possible tools are those that transform UML into code. This would allow to get implementations from the designs which gives many advantages to generate code from the designs. It definitely integrates aspects of classical computing with some quantum aspects.

3. (2020) [17] *Software engineering for 'quantum advantage'*:

It does not apply to any of the Perez-Delgado life cycle stages. It is an initial work that proposes the monads pattern to model the semantics of (classical-quantum) hybrid components that appear in the systems. This position paper argues for the need for a proper quantum software engineering discipline benefiting from precise foundations and calculi, capable of supporting algorithm development and analysis.

4. (2020) [18] *Selection of quantum computing architecture using a decision tree approach*:

This paper helps the decision-making process in choosing the right software architecture based on the type of the quantum project (Quantum annealing, Quantum Simulation and Universal Quantum Computing). It can be useful in the *Requirements Analysis* stage.

5. (2020) [19] *Integrating quantum computing into workflow modeling and execution*:

This work introduced the Quantum modelling Extension (QUANTME) to model quantum circuit invocations in workflows to ease their orchestration with classical applications and showed how to ensure the executability on different workflow engines. It is one of the few works that covers the *Observability* and *Analysis* stages. Another important aspect is that it has specific constructors to model the pre-processing and post-processing tasks (Readout Error Mitigation). It also proposes a "Quantum Software Lifecycle".

6. (2020) [20] *Modeling Quantum programs: Challenges, initial results, and research directions:*

The metamodel and the example is a bit simple compared to other analyzed works. It focuses on the stages of *Requirements Analysis*.

7. (2021) [21] *Towards Model-Driven Quantum Software Engineering:*

The approach of this work is applicable to several stages of the life cycle because it is based on MDE. Among them *"Requirements Analysis"*, *"Architecture & Design"* since from the models it is possible to obtain designs or codes to be applied to other stages of the life cycle.

8. (2021) [23] *Quantum Software Models: The Density Matrix for Classical and Quantum Software Systems Design*:

The work proposes a design procedure that aligns classical and quantum software through Density Matrix. The main contribution is to have a mathematical model that covers both Classical Software Systems and Quantum Systems. This proposal contributes to the *Quantum-Classical Splitting* stage and to propose a mapping between the two models. Future research could apply this mathematical model to transform systems between Quantum and Classical.

9. (2022) [22] *A Model-Driven Framework for Composition-Based Quantum Circuit Design*:

The paper presents a metamodel of quantum circuits containing as main classes are *layer*, *Quantum-Circuit*, *Quantum-Operation* and *Composite-Quantum-Operation* that allows to compose elements in a layer. It also shows a Meta-model for quantum library, which describes how a quantum operation is defined, for an arbitrary or fixed number of qubits. This is very useful in terms of reusability because the defined operation is independent of the number of qubits it should act on. What is innovative about this approach is the composition-oriented modelling language capability that works at a high level of abstraction while hiding low-level implementation details. Another valuable addition is the generation of code from designs, which promises to reduce the development effort compared to existing tools. This applies to the stages of *Requirements Analysis* and *Quantum-Classical Splitting*, *Implementation* and even *Deployment*.

Finally, the application of MDE concepts also suggests the use of well-known model-based transformation tools for quantum circuit transformations to different representations.

10. (2022) [24] *Software Architecture for Quantum Computing Systems – A Systematic Review*:

This systematic review facilitates researchers and technologists to develop new hypotheses, find references and propose architecture-centric frameworks for a new generation of quantum software. It can be the basis for future research and classifies and orders many of the works analyzed in this review.

Another interesting aspect of the paper is that it provides an ”Overview of quantum architecting process and its activities” map that can be useful when modelling the architecture of a quantum system. The paper takes a global view of quantum systems so it cannot be placed in one of the stages of Perez Delgado's work in particular. However, it has many references to very useful primary works to deepen in some of the topics of quantum software design and quantum architectures design. This paper contributes to the *Architecture & Design* and *Quantum-Classical Splitting* stages.

11. (2023) [25] *Model-Driven Optimization for Quantum Program Synthesis with MOMoT:*

In the same way as the previous works of the same authors, this can be used in several stages. In particular in this work that addresses the issue of model-driven optimization could be applied in the stages of *Implementation* and *Testing* to evaluate alternative quantum circuits.

12. (2023) [26] *A Graph-Based Approach for Modelling Quantum Circuits*:

This work mainly contributes to the *Implementation* stage as it presents different ways of conceptualizing a quantum circuit. The proposal allows a transformation between circuit representations. It can also contribute to the *Observability* stage as it points to the measurement-based quantum computation model, where the program is represented as a graph of entangled qubits and measurement operations. Having a Metamodel can also be the basis for possible Model-to-code transformations.

13. (2023) [27] *A Taxonomic View of the Fundamental Concepts of Quantum Computing–A Software Engineering Perspective*:

This work mainly provides a (BOK) body of knowledge on the main concepts involved in Quantum computing. This helps with the *Requirement and Analysis* stage of a project involving quantum programming. On the other hand, it is an interesting work for students and teachers to use as a starting point for understanding quantum computing.

14. (2023) [28] *Formalization of Quantum Intermediate Representations for Code Safety:*

This paper contributes to the formalism and portability of some QIR operations which detects insecure code in programs. This approach is applicable on *Requirements Analysis* (portability) and *Testing* (automatic verification) stages.

15. (2023) [29] *ScaffML: A Quantum Behavioral Interface Specification Language for Scaffold:*

The approach of this work applicable on specification stages *Requirements Analysis* and validation of software in debugging and *Testing*.

16. (2024) [30] *SimuQ: A Framework for Programming Quantum Hamiltonian Simulation with Analog Compilation:*

In this work, ad-hoc optimization strategies are applied for each specific platform. This is useful in the stages of *Observability* and *Analysis* of the results of the executions.

4. Conclusion. - Quantum systems are emerging as an innovative frontier in computing, promising to revolutionize the way we process information and tackle complex computational problems. As we move into an era where computing power is a critical resource, the application of quantum principles in software development becomes increasingly

crucial. These advances not only broaden our perspectives on what is possible in computing, but also pose exciting challenges in terms of quantum software design, implementation, and optimization. In particular, the design of such systems is a very new discipline and there are no consolidated techniques, tools or methodologies. In this work "An annotated bibliography" on the design of Quantum applications was carried out. For this purpose, different works that deal with this topic were reviewed. For each reference we made a brief commentary and tried to answer - what it tries to model (Design Effort) and - in which stage of the life cycle of the development of a quantum system. Most of the papers are oriented towards Math Oriented and Conceptual Models. Regarding the latter, the works of Perez Delgado (UML Profile) and Gemeinhardt, Felix et all. that aim to work with MDE techniques where they provide model-to-code transformations stand out. This type of initiatives seems to us relevant to deepen the research. It can also be noted that most of the work is focused on the stages of *Requirements Analysis*, *Quantum-Classical Splitting*, *Design* and *Implementation*. There is a deficit in the stages of *Analysis* of results and *Observability*. The latter are less common in traditional developments.

Continuing the current research trajectory opens two promising avenues. Firstly, conducting a Systematic Literature Review (SLR) would formalize research questions, survey relevant papers, and synthesize results, providing a comprehensive overview of the existing literature in quantum software modelling. Secondly, identifying and analyzing existing gaps within the discipline and proposing innovative solutions would involve a thorough examination of the current state-of-theart, pinpointing areas lacking understanding, methodologies, or tools. This proactive approach would foster innovation and ensure that quantum software modelling remains at the forefront of scientific and technological advancements.

References

[1] B. Boehm W., "Software Engineering," *IEEE Transactions on Computers*, vol. C-25, pp. 1226–1241, 12 1976.

[2] W. F. Bauer and A. M. Rosenberg, "Software," in *Proceedings of the December 5-7, 1972, fall joint computer conference, part II on - AFIPS '72 (Fall, part II)*, (New York, New York, USA), p. 993, ACM Press, 1972.

[3] A. G. Kleppe, J. B. Warmer, and W. Bast, *MDA explained : the model driven architecture : practice and promise*. Addison-Wesley, 2003.

[4] S. J. Mellor, *MDA distilled : principles of model-driven architecture*. Addison-Wesley, 2004.

[5] O. Pastor and J. C. Molina, "Model-driven architecture in practice: A software production environment based on conceptual modeling," *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*, pp. 1–302, 2007.

[6] M. A. Nielsen and I. L. Chuang, "Quantum Computation and Quantum Information 10th Anniversary Edition," *Cambridge University Press*, 2010.

[7] E. Rieffel and W. Polak, "Quantum computing : a gentle introduction," p. 372, 2011.

[8] M. J. Everitt, M. J. C. Henshaw, and V. M. Dwyer, "Quantum Systems Engineering: A structured approach to accelerating the development of a quantum technology industry," in *International Conference on Transparent Optical Networks*, vol. 2016-August, 2016.

[9] S. P. Wang and E. Sakk, "Quantum algorithms: Overviews, foundations, and speedups," *2021 IEEE 5th International Conference on Cryptography, Security and Privacy, CSP 2021*, pp. 17–21, 1 2021.

[10] M. Piattini, G. Peterssen, R. Perez-Castillo, J. L. Hevia, M. A. Serrano, G. Hernandez, I. G. R. de Guzm´ an, C. A. Paradela, M. Polo, E. Murina,´ L. Jimenez, J. C. Marque´ no, R. Gallego, J. Tura, F. Phillipson, J. M.˜ Murillo, A. Nino, and M. Rodr˜ ´ıguez, "The Talavera manifesto for quantum software engineering and programming," in *CEUR Workshop Proceedings*, vol. 2561, 2020.

[11] B. Kitchenham and O. Brereton, "Systematic literature reviews in software engineering–a systematic literature review," *Elsevier*, 2009.

[12] C. Pons, R. S. Giandini, and G. Perez,´ *Desarrollo de software dirigido por modelos*. Editorial de la Universidad Nacional de La Plata (EDULP) / McGraw-Hill Educacion, 2010.´

[13] F. Leymann, "Towards a Pattern Language for Quantum Algorithms," 6 2019.

[14] C. A. Perez-Delgado and H. G. Perez-Gonzalez, "Towards a Quantum´ Software Modeling Language," *Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020*, pp. 442–444, 6 2020.

[15] R. Perez-Castillo and M. Piattini, "Design of classical-quantum systems´ with UML," *Computing*, vol. 104, pp. 2375–2403, 11 2022.

[16] M. A. Serrano, R. Perez-Castillo, and M. Piattini,´ *Quantum Software Engineering*. 2022.

[17] L. S. Barbosa, “Software engineering for ’quantum advantage’,” *Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020*, vol. 3, pp. 427–429, 6 2020.

[18] L. Nallamothula, “Selection of quantum computing architecture using a decision tree approach,” *Proceedings of the 3rd International Conference on Intelligent Sustainable Systems, ICISS 2020*, pp. 644–649, 12 2020.

[19] B. Weder, U. Breitenbucher, F. Leymann, and K. Wild, “Integrating quantum computing into workflow modeling and execution,” *Proceedings - 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing, UCC 2020*, pp. 279–291, 12 2020.

[20] S. Ali and T. Yue, “Modeling Quantum programs: Challenges, initial results, and research directions,” *APEQS 2020 - Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software, Co-located with ESEC/FSE 2020*, pp. 14–21, 11 2020.

[21] F. Gemeinhardt, A. Garmendia, and M. Wimmer, “Towards ModelDriven Quantum Software Engineering,” in *Proceedings - 2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering, Q-SE 2021*, 2021.

[22] F. Gemeinhardt, A. Garmend´ıa, M. Wimmer, and R. Wille, “A ModelDriven Framework for Composition-Based Quantum Circuit Design,”

[23] I. Exman and A. T. Shmilovich, “Quantum Software Models: The Density Matrix for Classical and Quantum Software Systems Design,” *Proceedings - 2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering, Q-SE 2021*, pp. 1–6, 3 2021.

[24] A. A. Khan, A. Ahmad, M. Waseem, P. Liang, M. Fahmideh, T. Mikkonen, and P. Abrahamsson, “Software Architecture for Quantum Computing Systems – A Systematic Review,” *Journal of Systems and Software*, vol. 201, 2 2022.

[25] F. Gemeinhardt, M. Eisenberg, S. Klikovits, and M. Wimmer, “ModelDriven Optimization for Quantum Program Synthesis with MOMoT,” in *Proceedings - 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS-C 2023*, 2023.

[26] D. Alonso, P. Sanchez, and B.´ Alvarez, “A Graph-Based Approach for´ Modelling Quantum Circuits,” *Applied Sciences 2023, Vol. 13, Page 11794*, vol. 13, p. 11794, 10 2023.

[27] R. Juarez-Ram´ ´ırez, C. X. Navarro, S. Jimenez, A. Ramırez, V. TapiaIbarra, C. Guerra-Garc´ıa, H. G. Perez-Gonzalez, and C. Fernandez-y´ Fernandez, “A Taxonomic View of the Fundamental Concepts of Quan-´ tum Computing–A Software Engineering Perspective,” *Programming and Computer Software*, vol. 49, pp. 682–704, 12 2023.

[28] J. Luo and J. Zhao, “Formalization of Quantum Intermediate Representations for Code Safety,” 2023.

[29] T. Jin and J. Zhao, “ScaffML: A Quantum Behavioral Interface Specification Language for Scaffold,” in *Proceedings - 2023 IEEE International Conference on Quantum Software, QSW 2023*, 2023.

[30] Y. Peng, J. Young, P. Liu, and X. Wu, “SimuQ: A Framework for Programming Quantum Hamiltonian Simulation with Analog Compilation,” *Proceedings of the ACM on Programming Languages*, vol. 8, 2024.

[31] M. Bandic, H. Zarein, E. Alarcon, and C. G. Almudever, “On Structured Design Space Exploration for Mapping of Quantum Algorithms,” *2020 35th Conference on Design of Circuits and Integrated Systems, DCIS 2020*, 11 2020.

[32] X. Zhou, S. Li, and Y. Feng, "Quantum Circuit Transformation Based on Simulated Annealing and Heuristic Search," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, pp. 4683–4694, 8 2019.

[33] D. Wecker and K. M. Svore, "LIQUi—¿: A Software Design Architecture and Domain-Specific Language for Quantum Computing," 2 2014.

[34] C. A. Perez-Delgado, "A quantum software modeling language,"´ *Quantum Software Engineering*, pp. 103–119, 10 2022.

Author contribution:

1. Conception and design of the study
2. Data acquisition
3. Data analysis
4. Discussion of the results
5. Writing of the manuscript
6. Approval of the last version of the manuscript

LMB has contributed to: 1, 2, 3, 4, 5 and 6.

AF has contributed to: 1, 2, 3, 4, 5 and 6.

JMS has contributed to: 1, 2, 3, 4, 5 and 6.

OP has contributed to: 1, 2, 3, 4, 5 and 6.

Acceptance Note: This article was approved by the journal editors Dr. Rafael Sotelo and Mag. Ing. Fernando A. Hernández Gobertti.