Towards Process Centered Architecting for Quantum Software Systems

Aakash Ahmad*, Arif Ali Khan†, Muhammad Waseem‡, Mahdi Fahmideh§, Tommi Mikkonen¶
*College of Computer Science and Engineering, University of Ha'il, Saudi Arabia
†M3S Empirical Software Engineering Research Unit, University of Oulu, 90570 Oulu, Finland
‡School of Computer Science, Wuhan University, Wuhan, China
§School of Business at University of Southern Queensland, Queensland, Australia
¶Faculty of Information Technology, University of Jyväskylä, FI-40014 Jyväskylä, Finland
a.abbasi@uoh.edu.sa, arif.khan@oulu.fi, m.waseem@whu.edu.cn, mahdi.fahmideh@usq.edu.au, tommi.j.mikkonen@jyu.fi

***Abstract*—Quantum Software Engineering (QSE) is a recent trend - focused on unifying the principles of quantum mechanics and practices of software engineering - to design, develop, validate, and evolve quantum age software systems and applications. Software architecture for quantum computing (a.k.a. quantum software architectures (QSA)) supports the design, development, and maintenance etc. phases of quantum software systems using architectural components and connectors. QSA can enable quantum software designers and developers to map the operations of Qubits to architectural components and connectors for implementing quantum software. This research aims to explore the role of QSAs by investigating (i) architectural process having architecting activities, and (ii) human roles that can exploit available tools to automate and customise architecture-centric implementation of quantum software. Results of this research can facilitate knowledge transfer, enabling researchers and practitioners, to address challenges of architecture-centric implementation of quantum software systems.**

***Index Terms*—Quantum Software Engineering, Quantum Software Architecture, Architecture Process, Reference Architecture**

I. Introduction

Quantum software systems exploit quantum programming languages - implementing quantum algorithms - that enable the development, execution, and/or simulation of software applications on quantum computing platforms [1] [2]. Quantum programming languages enable software engineers and developers to write source code that manipulates quantum bits (Qubits) to control quantum gates (Qugates) for operationalising quantum computing systems [3]. Quantum software engineering has recently emerged as an engineering paradigm to design, develop, test, and evolve software systems that require quantum information processing such as quantum key distribution, quantum search systems, and quantum simulation [2]. Academic research [4] as well as industrial developments (e.g., Q#: Microsoft, Qiskit: IBM, and Cirq: Google), have promoted engineering and development of quantum age software solutions and scaled up strategic investments in quantum computing platforms [5]. Quantum programming languages (e.g., Q#, Qiskit), implementing quantum algorithms, are considered as a foundation for developing quantum software [1] [3]. However, such programming languages are designed to mainly focus on specifying the source code that produces executable specifications but undermines the overall global view of software systems in terms of architectural components (i.e., units of computation and storage) and connectors (i.e., interconnections between components).

Quantum Software Architecture (QSA) represents a class of software architectures, aiming to abstract implementation-specific details such as modules of source code and their interactions, represented as architectural components and their connectors for quantum software [6] [7]. QSAs have emerged as the most recent genre of software architectures and the current generation of software practitioners, i.e,. software engineers, architects, and developers find themselves less prepared to tackle architectural challenges of quantum software [2] [4] [6]. QSA challenges can include aspects of quantum domain engineering, quantum co-design (mapping Qugates to Qubits and their representation as architectural components), validation, deployment, and simulation of quantum software applications [2] [7]. In recent years, a number of reference architectures [6] and architectural models have been developed for QSAs [7]. However, there is no research on the notion of an architectural process, where the process acts as an umbrella to cover architecting activities, incorporate tool support to enable automation, and define professional roles for human decision support in quantum software development. This research aims to answer two research questions (RQs).

> **RQ-1**: What architecting process(es) and activities are proposed to develop quantum software?

RQ-1 aims to derive an architectural process, encapsulating a multitude of architecting activities for QSAs to enable a systematic and incremental architectural development of quantum software [8]. From a conceptual perspective, the existence of a process answers *what needs to be done*?, while the activities in a process focus on *how it is to be done*? [9].

> **RQ2**: Are there any human roles and tool support for architecting quantum software?

RQ-2 aims to investigate human roles (software practationers) that incorporate human decision in the process and

available tool support (enabling process automation).

Research method and contributions: This research focused on investigating a collection of published solutions on QSA, qualitatively selecting 32 research studies via a systematic review process [10] [11], also following the guidelines from our earlier conceptual modeming effort [12], to investigate RQ-1 and RQ-2. By following the systematic review approach, we analysed reference architectures, frameworks, and architectural implementations of quantum software systems and identified a total of 5 activities and organised them in a unified process, in line with the academic findings [8] [13] and industry-based studies on architectural processes [9]. To complement the process, human decision support and automation are also investigated and we identified 4 human roles specific to quantum software architecting and 11 tool prototypes that can enable architecture process automation. The architectural process and its underlying architecting activities are demonstrated with a case study on quantum key distribution architecture. We outline the primary contributions of this research as:

- Derivation of a process - identifying 5 architecting activities - that support a process-centered and incremental architecting (i.e., design, implementation, validation, and deployment) of quantum software.
- Identification of 4 professional roles and highlighting 11 available tools that can enrich the architecting process with human decision support and automation.

II. Background: Architecture for Quantum Software Systems

A. Software Systems for Quantum Computing

Quantum computers represent a paradigm shift from classical computing that relies on binary gates having [0, 1] binary digits, representing **On** and **Off** states to manipulate a digital circuit. Quantum gates that control quantum hardware are managed by quantum bits expressed as $\mid 0 \rangle$ and $\mid 1 \rangle$. Additional details about Qugates and Qubits are provided in [11]. Specifically, in Qubits, the state 0 is expressed as $\mid 0 \rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and the state 1 is expressed as $\mid 1 \rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

As in Figure 1, the hardware layer represents the most primitive element of quantum computing that is controlled by the software layer above it. The architecture of quantum software enables abstraction of quantum source code with architectural components and connectors during system design and implementation. In order to manage and control Qugates, quantum software systems rely on quantum compilers, acting as an intermediary between Qugates (hardware) and Qubits (software instructions) to compile quantum source code.

B. Quantum Software Architecture

Software architecture follows ISO/IEC/IEEE 42010:2011 standard to enable software designers and architects to implement software systems incrementally, i.e., design components and connectors to be translated into executable source code that can be tested, and evolved using architectural models,

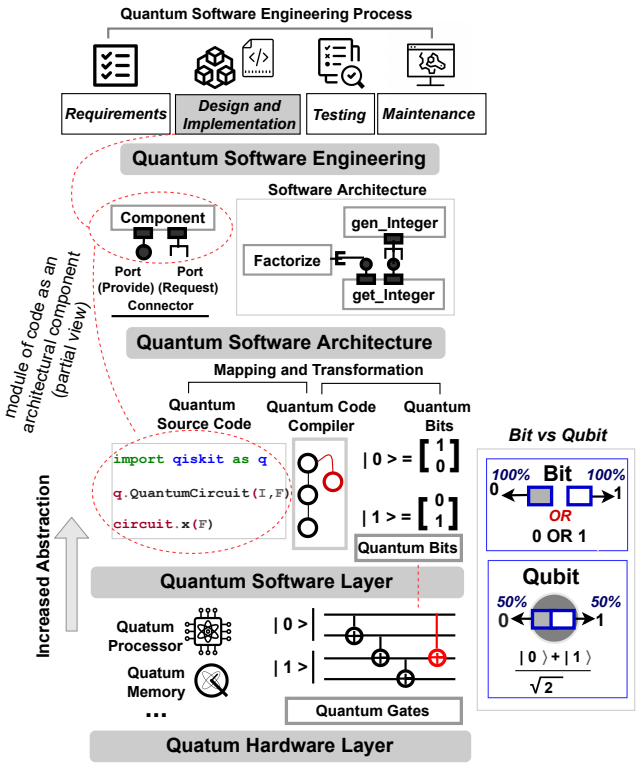

Fig. 1: A Layered Overview of Quantum Computing Systems

tools, and notations [8] [9]. Architecture for quantum software systems and applications empowers the role of architects to abstract the complexities of source code modules and their interactions as architectural components and connectors [7]. As in Figure 1, to develop a quantum search algorithm, a partial architectural view represents components for factorization (Factorize) of a generated integer (Gen_Integer). Architectural view helps to reflect an overall representation of the systems, elements of computation (components) and how they interact (connectors).

III. Process and Architecting Activities (RQ-1)

A. Architectural Process for Quantum Software

Academic research [8] and industry based studies [9] on architecture-centric software engineering have highlighted three generic architecting activities namely (i) architectural analysis, (ii) architectural synthesis, and (iii) architectural evaluation [14] [15]. From QSA perspective, generic activities need to be extended with a fine-granular process representation having specific architecting activities, in Figure 2, addressing quantum aspects of software systems.

The Quantum Aspects of Software Architecting: One of the limitations of existing architectural process(es) such as [8] [13], addressing design and development of traditional software systems, is the lack of support for requirements specific to QSE lifecycle. This means that software designers and developers who use QSE principle and practices to design quantum software need to design QSAs and/or develop the

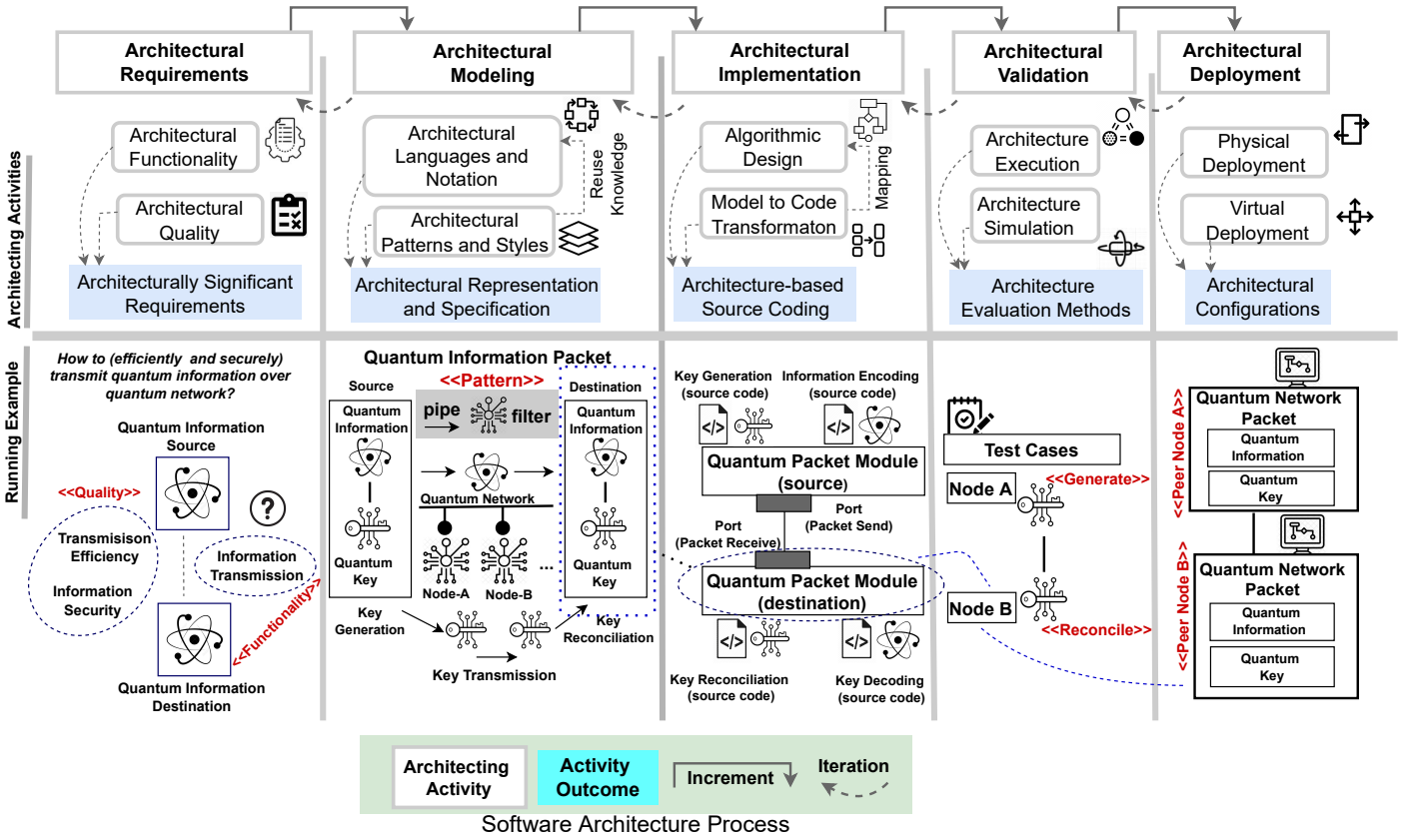

Fig. 2: Overview of the Architecture Process, Architecting Activities, and Demonstrative Example of QKD

underlying quantum algorithms attuned to operationalising quantum bits rather than classical binary digits [1] [3].

Deriving the Architectural Process and Activities: Process encapsulates a number of steps, referred to as activities, empowering the roles of designers and developers to utilise individual activities for specifying structure and behavior of software-intensive systems via architectural representation [9] [16]. To exemplify, consider the process in Figure 2, an activity named Architectural Requirements can outline the design challenges to be resolved by the architecture in terms of required software functionality and desired quality of software under development. More specifically the outlined ASR in running example: *how to effectively and securely transmit quantum information over quantum network?* highlights a design challenge, requiring architectural model to enable transmission of quantum information (required functionality) in an efficient and secure manner (desired quality).

B. Process Centred Architecting: A Demonstrating Example

Process centered architecting of quantum software, driven by architecting activities, is demonstrated with an incremental design and development of Quantum Key Distribution (QKD) solution (Figure 2). The QKD solution [17] enables secure processing and transmission of quantum information by generating and distributing symmetric cryptographic keys between two geo-distributed nodes utilising the principles of quantum physics. We introduce each activity and detail the architectural design aspect each activity supports (e.g., architectural validation) with an example case of QKD.

- **Activity I - Architectural Requirements (AR)**: as the initial activity of the process, AR aims to identify and specify required functionality and desired quality (a.k.a. functional and quality constraints) of the software.

  Example: The architectural requirement(s) specify the needs for quantum information processing and transmission over quantum network of two nodes referred to as source (the sender) and destination (the receiver). AR is formulated as: *'how to (securely and efficiently) transmit quantum information over quantum network'*.

  Activity Outcome: is a set of ASRs to streamline the required *functionality*, i.e., transmission of quantum information along with the desired *quality* that complements functionality with efficient and secure transmisison.
- **Activity II - Architectural Modeling (AM)**: relies on ASR to create a model of the architecture for visual representation or specification of functionality and quality of the software. Quantum modeling notations such as QML [15] or architectural languages [9] can be used for semi-formal representation of the model.

  Example: Modeling of QKD based on ASR indicates that a pattern named Pipe and Filter architecture [8] is being

applied to enable processing of quantum information (desire functionality) across network nodes. Quantum key generation and key reconciliation mechanisms are used to ensure efficient and secure communication.

Activity Outcome: is an architectural model and its underlying specifications (pattern based solution) that act as a blue-print to implement the QKD solution.

- **Activity III - Architectural Implementation (AI)**: focuses on designing algorithms and writing source code that adhere to the architectural representation and specification [1]. A quantum programming language [3] enables source coding of quantum software, whereas model-driven architecting can enable (semi-) automated generation of source code from architectural specifications [18].
  Example: Architectural component (Quantum Packet Module) as an element of computation abstracts details of source code modules and architectural connector (Packet Send/Receive) that represents module interconnection.
  Activity Outcome: is source code that can be executed or simulated on quantum computing platforms. Source coding provides concrete specifications of the ASR implemented via architectural modeling.
- **Activity IV - Architectural Validation (AV)**: executes or simulates the source code to validate the ASRs and verify if the modeled and implemented architecture conforms to the needed functionality and quality. Architectural validation is conducted by means of architectural evaluation methods [13] (e.g., Software Architecture Analysis Method (SAAM)) to objectively evaluate the architecture and test for faults and errors in quantum source code [19].
  Example: In addition to architectural evaluation using SAAM or alike methods, source code testing is vital for generating and validating the test code for quantum key generation and quatum key recociliation.
  Activity Outcome: is to validate architecture model against ASRs and source code to be deployed and executed on quantum computing platforms.
- **Activity V - Architectural Deployment (AD)**: as the last activity supports deployment of source code modules on quantum computing platforms. Deployment can be physical (code executed on quantum hardware) or virtual (code simulated on hybrid or non-quantum platform) [20]. Supporting either of these two deployments, architectural process completes a cycle enabling an incremental development from requirements to deployment [21] [22].
  Example: Architectural components for QKD are deployed on two peer-nodes that can support secure transmission of quantum information via QKD.
  Activity Outcome: is architectural configurations managed as source code libraries (e.g., packages, APIs, modules) that can be replicated and executed on quantum computing (nodes) after architectural validation.

IV. Human Roles and Tool Support (RQ-2)

QSAs as an emerging genre of software-intensive systems require unique expertise such as quantum domain engineering (mapping quantum hardware and software), quantum software architecting (translating Qubits to architectural components), and quantum code simulation (analysing flow of quantum information processing) as in Figure 3.

A. Quantum Domain Engineers

The role of quantum domain engineer is rooted into domain-specific software engineering [8], however, in a quantum domain, the domain engineer needs to analyse quantum-specific attributes like mapping between Qugates and their corresponding Qubit representation [18].

- **Available Tool Prototype(s)**: Auto E/E Framework [23], Link Layer [24]. We identified the above-mentioned tool prototypes that support the role of quantum domain engineers as in Figure 3. For example, the tool prototype named Auto E/E framework [23] automates architectural requirement activities to bridge the gap between quantum hardware of an embedded system and the instruction set (source code specifications) to enable quantum information processing in automotive domain.
- **Process Support**: Quantum domain engineers can help software designers to map architectural requirements (e.g., mapping components to Qubits) to create an architectural model for quantum software implementation.

B. Quantum Software Architects

The role of software architect in QSE is to oversee the architectural process. An architect is responsible for identifying the design problems and planning the development of a software system that in turn satisfies functional requirements [25].

- **Available Tool Prototype(s)**: Strawberry Fields [26] an open source tool is developed for quantum software designers and architects to design and optimise the software applications for photonic quantum computers. Strawberry Fields automates the design activities by converting the domain specific code model to be executeed using photonic quantum computer.
- **Process Support**: Quantum software architects use quantum simulation tools for modeling the architectural components as in Figure 3.

C. Quantum Code Developers

Quantum code developers utilise architectural modeling to develop quatum source code for architectural implementation [14]. Developers (both algorithm designers, code developers) require knowledge about software tool chains in QSE lifecycle.

- **Available Tool Prototype(s)**: Various tools are available to assist the quantum software developers in performing the development activities. For example, XACC (eXtremescale ACCelerator) [27] is used to compile the code generated using both quantum and classical programming languages, Figure 3, independent of programming frameworks, hardware, and computational models.

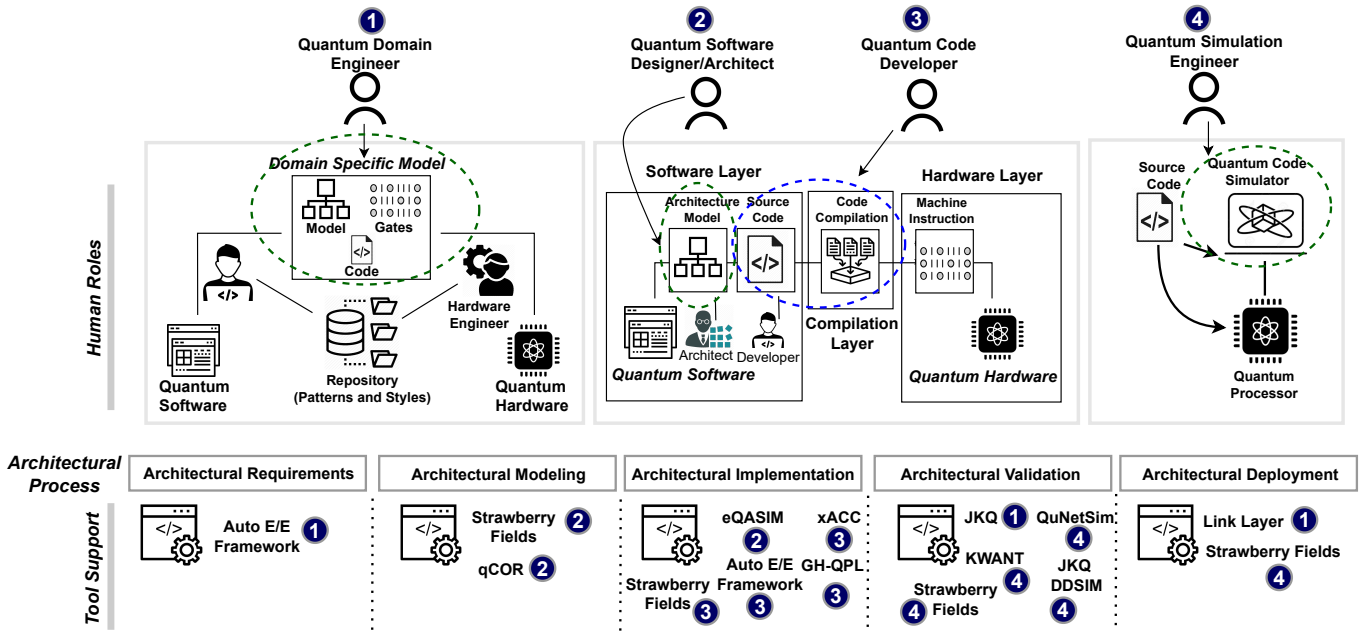

Fig. 3: Human Roles and Tool Support in Architecting Activities

- **Process Support**: The architecture implementation activities are supported by the quantum software developers. They define the implementation strategies of software architecture at the intersection of quantum hardware, quantum-classical compilers, and programming models.

D. Quantum Simulation Engineer

Quantum simulation engineer manages execution and simulation of quantum software, validating the quality, ensuring that the developed application fits the purpose and is error free.

- **Available Tool Prototype(s)**: QuNetSim [28] simulator is developed to design and test the robustness of the quantum network protocols. It helps the simulation engineer to test quantum network applications tasks developed to transmit and store the quantum information.
- **Process Support**: Quantum simulation engineer supports both architectural validation and deployment activities. The process support involves implementing testing frameworks, bug triage, enhancing reliability efforts by improving testing infrastructure, and supporting open source projects like error correction libraries.

V. Related Work

A. Engineering and Development Life-cycles for QSW

Increased adoption of quantum computing systems and infrastructures in industrial, as well as organisational context [5] [29], has resulted in the application of software engineering (SE) methods and techniques - referred to as Quantum Software Engineering (QSE) - to develop quantum software applications [2] [4]. Traditional SE that focuses on processes and practices has been tailored to address the challenges of QSE. Specifically, the research in [30] takes into account the concepts of quantum computing (e.g., Qugates and Qubits) to propose a software development lifecycle (SDLC) for quantum software. The proposed SDLC enables quantum software designers and architects to incorporate the concepts of (quantum) domain engineering to model quantum circuits that act as the foundation for implementing quantum algorithms. Quantum specific SDLCs [30] [2] address some of the challenges for QSE such as quantum domain engineering [6], quantum information simulation and validation [19]. However, such development life-cycles act as reference models and lack support for specifications (source coding), tools (automation), and patterns (best practices) for software implementation [14].

B. Software Architecture Solutions for Quantum Computing

Software design is an integral phase of SE that abstracts away implementation details by establishing an architectural blueprint for the structure and behavior of software systems [8] [9]. In the context of QSE [18], a recently conducted mapping study highlights the role of architectures in the development of quantum software [7]. Quantum Software Architecture (QSA) solutions such as [16] exploit architectural models to transform a quantum algorithm from high-level models to concrete executable specifications. Specifically, the research exploits layered architecture pattern [14] [15] to design the quantum algorithm, implement it, and simulate it in an incremental manner using software layering. In similar research on QSA-driven implementation of quantum software, the researchers in [22] have proposed software architecture and presented the design flow to compile a quantum program from high-level host language to hardware-specific instructions.

TABLE I: Roles and Tool Support in Architecture Process

Human Roles	✓: Role Involved, 1-11: ID of Tool				
	Architecture Process and Activities				
	AR	AM	AI	AV	AD
Quantum Domain Engineer	✓**1**			✓**7**	✓**10**
Quantum Software Architect		✓**2, 3**	✓**4**		
Quantum Code Developer			✓**1, 2** ✓**5, 6**		
Quantum Simulation Engineer				✓**2, 8** ✓**9, 11**	
Tools Prototypes					
1 = Auto E/E Framework. 2 = Strawberry Fields. 3 = qCOR. 4 = eQASIM. 5 = xACC. 6 = GH-QPL. 7 = JKQ. 8 = QuNetSim. 9 = KWANT. 10 = Link Layer 11 = JKQ DD SIM.					

VI. Conclusions

QSE as an emergent class of software engineering aims to apply principles and practices of software design and development to engineer quantum algorithms that can be executed/simulated on quantum platforms. The focus of this research is overviewed in Table I - acting as a structured catalogue - that maps architecting activities (Section III), professional roles, and available tool support (Section IV) to enrich QSA process. For example, Table I highlights that the role of *Quantum Domain Engineer* enables managing (i) ASRs and supporting (ii) architectural deployment activities in QSA. Primary contributions of this research are:

- Streamlining the architectural process, professional roles, and available tool prototypes to complement QSE by enabling quantum software development via QSAs.
- Architectural process acts a reference model to guide researchers and practitioners to utilise or customize existing architectural processes, patterns, tools, and human expertise to design emerging and next generation QSAs.

References

[1] F. T. Chong, D. Franklin, and M. Martonosi, "Programming languages and compiler design for realistic quantum hardware," *Nature*, vol. 549, no. 7671, pp. 180–187, 2017.

[2] J. Zhao, "Quantum software engineering: Landscapes and horizons," *arXiv preprint arXiv:2007.07047*, 2020.

[3] M. Ying, *Foundations of quantum programming*. Morgan Kaufmann, 2016.

[4] M. Piattini, M. Serrano, R. Perez-Castillo, G. Petersen, and J. L. Hevia, "Toward a quantum software engineering," *IT Professional*, vol. 23, no. 1, pp. 62–66, 2021.

[5] R. Courtland, "Google aims for quantum computing supremacy [news]," *IEEE Spectrum*, vol. 54, no. 6, pp. 9–10, 2017.

[6] P. Murali, N. M. Linke, M. Martonosi, A. J. Abhari, N. H. Nguyen, and C. H. Alderete, "Full-stack, real-system quantum computer studies: Architectural comparisons and design insights," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pp. 527–540, IEEE, 2019.

[7] 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," *arXiv preprint arXiv:2202.05505*, 2022.

[8] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, "A general model of software architecture design derived from five industrial approaches," *Journal of Systems and Software*, vol. 80, no. 1, pp. 106–126, 2007.

[9] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 869–891, 2012.

[10] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep. EBSE Technical Report EBSE-2007-01, Keele University and Durham University, 2007.

[11] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya, "Quantum computing: A taxonomy, systematic review and future directions," *Software: Practice and Experience*, vol. 52, no. 1, pp. 66–114, 2022.

[12] M. Fahmideh, A. Ahmed, A. Behnaz, J. Grundy, and W. Susilo, "Software engineering for internet of things: The practitioner's perspective," *arXiv preprint arXiv:2102.10708*, 2021.

[13] Z. Li, P. Liang, and P. Avgeriou, "Application of knowledge-based approaches in software architecture: A systematic mapping study," *Information and Software technology*, vol. 55, no. 5, pp. 777–794, 2013.

[14] N. Khammassi, I. Ashraf, J. Someren, R. Nane, A. Krol, M. A. Rol, L. Lao, K. Bertels, and C. G. Almudever, "Openql: A portable quantum programming framework for quantum accelerators," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 18, no. 1, pp. 1–24, 2021.

[15] R. Pérez-Castillo, L. Jiménez-Navajas, and M. Piattini, "Modelling quantum circuits with uml," in *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, pp. 7–12, IEEE, 2021.

[16] K. M. Svore, A. V. Aho, A. W. Cross, I. Chuang, and I. L. Markov, "A layered software architecture for quantum computing design tools," *Computer*, vol. 39, no. 1, pp. 74–83, 2006.

[17] M. Mehic, M. Niemiec, S. Rass, J. Ma, M. Peev, A. Aguado, V. Martin, S. Schauer, A. Poppe, C. Pacher, *et al.*, "Quantum key distribution: a networking perspective," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–41, 2020.

[18] F. Gemeinhardt, A. Garmendia, and M. Wimmer, "Towards model-driven quantum software engineering," in *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, pp. 13–15, IEEE, 2021.

[19] P. Zhao, J. Zhao, Z. Miao, and S. Lan, "Bugs4q: A benchmark of real bugs for quantum programs," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1373–1376, IEEE, 2021.

[20] F. T. Chong, D. Franklin, and M. Martonosi, "Programming languages and compiler design for realistic quantum hardware," *Nature*, vol. 549, no. 7671, pp. 180–187, 2017.

[21] B. Sodhi and R. Kapur, "Quantum computing platforms: assessing the impact on quality attributes and sdlc activities," in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, pp. 80–91, IEEE, 2021.

[22] T. Häner, D. S. Steiger, K. Svore, and M. Troyer, "A software methodology for compiling quantum programs," *Quantum Science and Technology*, vol. 3, no. 2, p. 020501, 2018.

[23] H. Lan, C. Zhang, and H. Li, "An open design methodology for automotive electrical/electronic system based on quantum platform," *Advances in Engineering Software*, vol. 39, no. 6, pp. 526–534, 2008.

[24] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpedek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, *et al.*, "A link layer protocol for quantum networks," in *Proceedings of the ACM Special Interest Group on Data Communication*, pp. 159–173, 2019.

[25] K. M. Svore, A. V. Aho, A. W. Cross, I. Chuang, and I. L. Markov, "A layered software architecture for quantum computing design tools," *Computer*, vol. 39, no. 1, pp. 74–83, 2006.

[26] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy, and C. Weedbrook, "Strawberry fields: A software platform for photonic quantum computing," *Quantum*, vol. 3, p. 129, 2019.

[27] A. J. McCaskey, E. F. Dumitrescu, D. Liakh, M. Chen, W.-c. Feng, and T. S. Humble, "A language and hardware independent approach to quantum–classical computing," *SoftwareX*, vol. 7, pp. 245–254, 2018.

[28] S. DiAdamo, J. Nötzel, B. Zanger, and M. M. Beşe, "Qunetsim: A software framework for quantum networks," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–12, 2021.

[29] S. Vernacchia, "Quantum leap." https://www.pwc.com/m1/en/world-government-summit/documents/wgs-quantum-leap.pdf. Accessed: 2022-02-15.

[30] B. Weder, J. Barzen, F. Leymann, M. Salm, and D. Vietz, "The quantum software lifecycle," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software*, pp. 2–9, 2020.