




# Towards View-based Development of Quantum Software

Joshua Ammermann <sup>1</sup>, Wolfgang Maurer <sup>2</sup>, and Ina Schaefer <sup>1</sup>

**Abstract:** Quantum computing is an interdisciplinary field that relies on the expertise of many different stakeholders. The views of various stakeholders on the subject of quantum computing may differ, thereby complicating communication. To address this, we propose a view-based quantum development approach based on a Single Underlying Model (SUM) and a supporting quantum Integrated Development Environment (IDE). We highlight emerging challenges for future research.



**Keywords:** Quantum Computing, View-based Development, Integrated Development Environment


## 1 Introduction

Quantum computing is an interdisciplinary field that draws upon expertise ranging from the concrete quantum mechanical effects to the systematic development of quantum software. Many different stakeholders, each with a distinct background, are involved, such as mathematicians, physicists, engineers, and computer scientists. The mental models of stakeholders and their preferred view on quantum computing and programming may differ, which presents a significant challenge in terms of interdisciplinary collaboration. Stakeholders want to work with their preferred view of a quantum program while being supported by a model- and view-based approach that combines mutual views.

There are some established concepts and tools that support one or more views of quantum computing. Such views include mathematical formulas (linear algebra), quantum circuits, and Quantum Programming Languages (QPLs). Many QPLs are embedded in a classical host language (e.g., Qiskit is embedded in Python), so they can reuse its tooling. For other languages such as Q# [Sv18] a Visual Studio Code plugin is available as part of the Azure Quantum Development Kit<sup>3</sup>. While this tooling provides comfortable editors (e.g., with syntax highlighting), it lacks support for adequate quantum views other than the QPL. Q-UML [Pé22] extends the Unified Modeling Language (UML) with quantum elements, such as quantum classes or variables, but fails to address quantum-specific characteristics adequately [AY23]. First Integrated Development Environments (IDEs) for quantum software are under development. Quirk<sup>4</sup> and IBM Composer<sup>5</sup> both provide

---

<sup>1</sup> KIT, Institute of Information Security and Dependability (KASTEL), Karlsruhe, Germany, [joshua.ammermann@kit.edu](mailto:joshua.ammermann@kit.edu),  <https://orcid.org/0000-0001-5533-7274>;  
[ina.schaefer@kit.edu](mailto:ina.schaefer@kit.edu),  <https://orcid.org/0000-0002-7153-761X>

<sup>2</sup> Technical University of Applied Sciences Regensburg and Siemens Technology, Regensburg/Munich, Germany, [wolfgang.maurer@othr.de](mailto:wolfgang.maurer@othr.de),  <https://orcid.org/0000-0002-9765-8313>

<sup>3</sup> <https://github.com/microsoft/qsharp>

<sup>4</sup> <https://github.com/Strilanc/Quirk>

<sup>5</sup> <https://quantum.ibm.com/composer>

a circuit editor, while IBM Composer has an additional OpenQASM / Qiskit view, but only supports one-way-editing for Qiskit code. The QuantumPath IDE [HPP22] provides disjoint annealing, circuit, and code editors. The Classiq IDE [Mi22] provides a high-level description in a quantum modeling language that can be synthesized into circuits, but this is merely a unidirectional transformation. Thus, existing IDEs still mostly support only one-way-editing / unidirectional transformations between views. These are not sufficient to support a collaborative workflow between an interdisciplinary group of stakeholders.

In classical software engineering, a model is an abstraction of a real system towards specific aspects of interest. Model-based techniques use models as the central artifact, facilitating the creation of domain-specific views and languages [WB23]. View-based development techniques allow the organization and generation of different views through the use of a (Virtual) Single Underlying Model (V-SUM) [Me20]. A V-SUM ensures consistency of underlying model(s) and derived views, and enables collaborative editing. These techniques are being adopted in interdisciplinary fields, such as cyber-physical systems [Re23], and appear suitable for addressing the needs of the interdisciplinary field of quantum computing.

A comprehensive approach to view-based development of quantum software is yet to be established. Such an approach enables stakeholders to view and edit different views, which will be held consistent automatically. Thus, the research challenge is: How can a view-based approach to quantum software development be defined and realized?

## 2 View-based Quantum Development Approach

We propose a View-based Quantum Development approach based on the V-SUM to comprise different views of quantum computing suitable for stakeholders with varying backgrounds. Fig. 2 shows the approach based on the Quantum Single Underlying Model (Q-SUM), a V-SUM [Me20] tailored for quantum software, as the central artifact. This model contains cross-disciplinary information such as mathematical descriptions, quantum circuits, and program code. The Q-SUM allows information to be kept consistent between views, while quantum-specific views can be created dynamically via projection. Also, only the integration into the Q-SUM is required, for instance, when adding support for a new QPL.

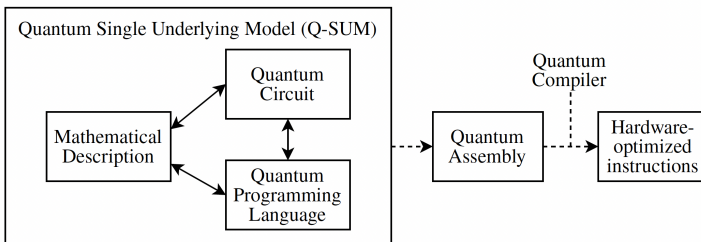


Fig. 1: View-based Quantum Development approach relying on a Q-SUM

Quantum Assembly (QASM) can be used as an interface to quantum compilers that handle optimization for the specific quantum hardware. This approach reuses existing concepts from view- and model-based development and takes advantage of their benefits.

### 3 Quantum IDE for the View-based Quantum Development Approach

A View-based Quantum Development approach requires tool support in order to be applicable in practice. We propose a Quantum IDE based on the Q-SUM for the collaborative view-based development of quantum software. Fig. 2 depicts how we envision such an IDE. A rich circuit editor ① and QPL editor ② should be provided for editing of quantum circuits and quantum program code respectively. Our IDE addresses open research questions in quantum software identified by [AY23]: encapsulation, abstraction, modularity, and quantum platform independence. Standard gates and more complicated building blocks of quantum software (e.g., a diffusion operator) could be accessed from a library ③. The creation of user-defined blocks and the composition of building blocks should also be possible. An inspector ④ should provide a mathematical description and allow configuration of the mapping of connections (e.g., quantum variables as parameters) for a selected building block. The execution ⑤ of the modeled quantum software on quantum hardware and the simulation and inspection of the system state at different points in time should be supported. For this constraints on the simulation and execution of quantum software must be considered. The IDE vision is a representative example based on currently employed means of abstraction, but finding appropriate abstractions is an important part of the endeavor. For example, one might consider a diagrammatic language [Co10] alternatively to the circuit representation.

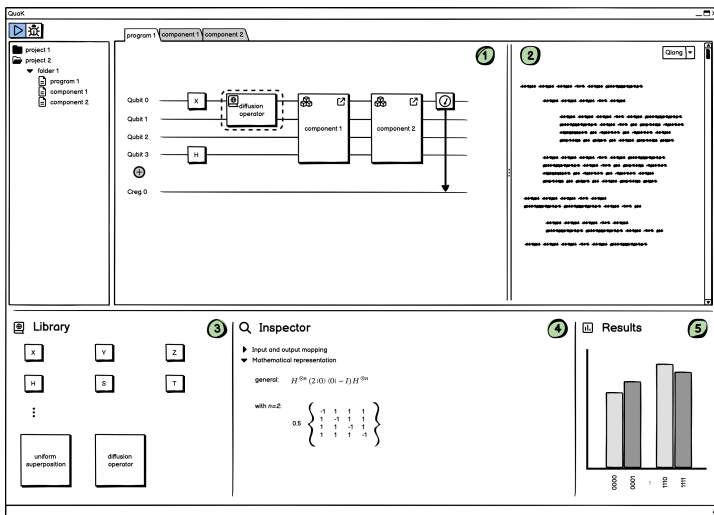


Fig. 2: Vision of an IDE for the View-based Quantum Development approach

## 4 Research Challenges

Realizing this View-based Quantum Development approach with a supporting IDE will require solving the following research challenges: The conceptual challenge is the design of the Q-SUM that manages the consistency of different views of quantum software based on existing view- and model-based technologies. To design such a model, it is necessary to identify and analyze the various possible views, and their respective benefits and drawbacks. The design has to be extensible to integrate existing and future QPLs and support design patterns, best practices, modularity, and compositionality. Implementation challenges are the design and implementation of the IDE using a selection of state-of-the-art tooling, and the integration of external quantum simulators and interfaces to quantum hardware. For the evaluation of the Q-SUM, use case scenarios have to be defined and the capabilities of the approach have to be evaluated against them. Finally, a user study with domain experts should be conducted to rate the usability of our approach and its supporting IDE.

## Acknowledgements

This work has been supported by the German Ministry for Education and Research in project QuBRA under Grant No.: 13N16303, and in TAQO-PAM under Grant No.: 13NI6092.

## References

- [AY23] Ali, S.; Yue, T.: On the Need of Quantum-Oriented Paradigm. In: QP4SE 2023. ACM, URL: <https://doi.org/10.1145/3617570.3617868>, 2023.
- [Co10] Coecke, B.: Quantum Picturalism. *Contemporary Physics* 51 (1), DOI: 10.1080/00107510903257624, arXiv: 0908.1787, 2010.
- [HPP22] Hevia, J. L.; Peterssen, G.; Piattini, M.: QuantumPath: A Quantum Software Development Platform. *Software: Practice and Experience* 52 (6), DOI: 10.1002/spe.3064, 2022.
- [Me20] Meier, J. et al.: Classifying Approaches for Constructing Single Underlying Models. In: *Model-Driven Engineering and Software Development*. Vol. 1161, Springer, DOI: 10.1007/978-3-030-37873-8\_15, 2020.
- [Mi22] Minerbi, N.: Quantum Software Development with Classiq. In: *Quantum Software Engineering*. Springer, DOI: 10.1007/978-3-031-05324-5\_14, 2022.
- [Pé22] Pérez-Delgado, C. A.: A Quantum Software Modeling Language. In: *Quantum Software Engineering*. Springer, DOI: 10.1007/978-3-031-05324-5\_6, 2022.
- [Re23] Reussner, R. et al.: Consistency in the View-Based Development of Cyber-Physical Systems (Convide). In: *MODELS-C 2023*. IEEE, DOI: 10.1109/MODELS-C59198.2023.00026, 2023.
- [Sv18] Svore, K. et al.: Q#: Enabling Scalable Quantum Computing and Development with a High-level DSL. In: *RWDSL 2018*. ACM, DOI: 10.1145/3183895.3183901, 2018.
- [WB23] Wąsowski, A.; Berger, T.: *Domain-Specific Languages: Effective Modeling, Automation, and Reuse*. Springer, ISBN: 978-3-031-23668-6, 2023.