Towards Model-Driven Quantum Software Engineering

Felix Gemeinhardt  
Johannes Kepler University  
Linz, Austria  
Email: felix.gemeinhardt@jku.at

Antonio Garmendia  
Johannes Kepler University  
Linz, Austria  
Email: antonio.garmendia@jku.at

Manuel Wimmer  
Johannes Kepler University  
Linz, Austria  
Email: manuel.wimmer@jku.at

Abstract—Quantum technologies are emerging. Dedicated languages for programming Quantum machines are emerging as well and already used in different settings. Orthogonal to this development, Model-Driven Engineering (MDE) is explored to ease the development of software systems by applying modeling techniques such as Domain-Specific Modeling Languages and generative techniques such as code generation.

In this position paper, we argue for a dedicated research line which deals with the exploration of how MDE may be applied for Quantum technologies. This combination would allow to speed-up the development of Quantum software, open the door for domain experts to utilize Quantum technologies, and may provide an additional abstraction layer over existing Quantum hardware architectures and programming languages. We outline several research challenges which we derived from a demonstration case of how to exploit domain-specific modeling for social network analysis on Quantum technologies.

I. INTRODUCTION

Nowadays, we face an increase in the use of Quantum Computing (QC) due to the launch of quantum programming languages (e.g., Microsoft Q# [1] or IBM’s Qiskit [2]) and the access to Quantum technologies offered by Cloud services. However, since QC is in the early stages, developers spent a lot of effort in the implementation of a QC application [3]. In this sense, we should of course apply the lessons learned from the evolution of classical programming, in which there was a development process until reaching an appropriate abstraction level such as offered by current object-oriented languages in order to efficiently develop software systems. It is reasonable to assume that QC must go through a similar process.

In order to support this process, it seems beneficial to apply Model-Driven Engineering (MDE) [4] to QC. MDE is a software development methodology that advocates for the use of models in all phases of software development. Models are either created with general-purpose modeling languages, e.g., UML [5], or with Domain-Specific Modeling Languages (DSMLs), i.e., languages specialized to an application domain. The construction of DSMLs starts with the definition of the abstract syntax, usually defined by a meta-model capturing the domain-specific concepts and their properties. DSMLs can be further categorized into languages focusing either on the problem domain (e.g., the one presented in Section 2) or the solution domain (e.g., easing the programming of a particular computing device such as QC technologies).

Fig. 1 shows a general overview of using models to create hybrid applications, i.e., applications that consists of the combination of classical and quantum computing technologies. The Meta-meta-model layer includes the definition of Meta-Object Facility (MOF) language, which permits the definition of meta-models [6]. The Meta-model layer contains the definition of languages that conforms to MOF, such as UML [5], the creation of custom DSMLs which focus on certain problem domains, or dedicated modeling languages for QC (QuantumML for short). The Model layer is where we find the models that are instances of meta-models. Based on such models, one may also create Hybrid Applications that require the combination of quantum and classical computers. Especially in the era of current NISQ devices, hybrid quantum/classical algorithms (like the ones we utilize in Section 2) promise to yield practical advantage compared to purely classical or purely quantum methods [7]. However, modeling languages have to account for this hybrid character, and therefore, be able to represent classical as well as QC elements. DSMLs for a given problem domain on the other hand require the ability to cope with classical, quantum as well as hybrid methods by employing novel code generators producing hybrid applications. The goal of this paper is to explore this opportunity.

Related Work. In order to achieve an abstraction level in the spirit of MDE, one has to decouple from the concrete hardware and software offered by vendors like IBM, Google, or D-Wave. In this respect, a collaborative quantum application platform has been proposed which includes, beside others, a hybrid-app repository and a quantum pattern repository [8].
Such a platform which shall not only include various algorithms, but also according data for proper usage, may benefit from the methods of MDE. Concerning the solution domain, a conceptual model for quantum programs and quantum states—which is independent of any modeling language, quantum programming language, and QC platforms—has been suggested by Ali & Yue [3], referring to the Quantum Models in Fig. 1. Although this suggestion concerns the solution domain, the authors consider quantum software modeling notations and methodologies a future challenge and expect the need for DSMs, which are tailored for modeling quantum software in problem domains such as physics or chemistry [3]. Concerning modeling notations, possible extensions to UML (called Quantum UML) for class diagrams and sequence diagrams have been presented [9]. Therefore, this approach may be categorized as a Quantum ML in Fig 1. A modeling extension for imperative workflow languages is introduced by Weder et al. [10], enabling the integration of quantum computations and the mapping of these extensions to native modeling constructs. The eXtreme-scale ACCelerator (XACC) [11] represents a hardware-agnostic execution framework for QC. The specified process allows conducting quantum circuits on various quantum architectures independent of the used language. Although there have been efforts concerning modeling quantum software in the solution space, to our knowledge there are no proposals on how to use problem-oriented DSMs for the application of hybrid quantum/classical methods, which are most promising to yield practical value in the near-term future. The following demonstration case shall serve as a first insight on what is necessary to port existing DSMs to the QC paradigm.

II. Demo Case: Social Network Analysis

In this section, we present a concrete demo case of model-driven optimization [12] using QC.

Modeling solution for SNA. Social Network Analysis (SNA) following an MDE approach requires the design of a meta-model that formalizes its structure based on graphs. Fig. 2 (label 1) shows the meta-model excerpt to represent social networks. This meta-model allows the addition of Persons and their Relationships. It permits also the definition of clusters (class Cluster), which comprises the selection of existing persons. By using this representation, community detection can be established by the creation of clusters based on analyzing the existing relationships between persons.

SNA solved by QC. There are many classical solutions to analyze SNA. However, in terms of efficiency, QC-based algorithms seem to be promising [13]: given the development of proper hardware. Fig. 2 (label 2) shows the meta-model excerpt to define the QC algorithms on the meta-model level. This meta-model can be instantiated to create a QuantumLibrary which contains a set of QuantumAlgorithms. There exist algorithms that just require the definition of the social network as input. However, others require additional data. For this purpose, QuantumAlgorithms may have Parameters.

Deployment on QC technology. In order to build the bridge between the social network and the QC meta-model we design the meta-model that it is shown in Fig. 2 (label 3). The main class of this meta-model is AlgorithmExecution which links the algorithms with the social network data. This meta-model represents a weaving model since it connects two different kinds of meta-models. By using this representation, we can deploy the algorithms in different QC technology and offer different deployment solutions.

Evaluation. In order to demonstrate the feasibility of our approach, we choose the Zachary Karate Club data-set [14]. Specifically, this data-set is a symmetric binary matrix of 34x34, which represents the existence or an absence of a link among persons. This matrix was transformed as an instance model compatible with the social network meta-model.

We make use of three algorithms within this demo case. The first algorithm is the quantum annealing based QA1 that utilizes the hybrid solver service for discrete quadratic models offered by D-Wave [15]. Secondly, a first gate-based hybrid algorithm (GB1) iteratively finds the strongest community in a given graph. Third, a second gate-based algorithm (GB2) recursively separates the graph into two communities. The model that contains these algorithms is shown in Fig. 2 as an instance of QuantumLibrary meta-model.

Finally, we define a QC deployment model as shown in Fig. 2. This model is the main artifact to deploy the hybrid application. To do this, we implemented a code generator based on Xtend [16].

III. Research Roadmap

We now present several research questions (RQs) leading to a research roadmap for the software engineering community interested in applying MDE, in particular, DSMs for QC.

RQ1: Modeling data structures and algorithms: Currently, we only model data structures and utilize algorithms which are already defined. However, a holistic MDE approach to QC has to consider transformations to appropriate QC data structures as well as a proper way of modeling quantum
algorithms themselves, as it has been proposed by Ali & Yue [3]. Especially in the era of current NISQ devices, hybrid algorithms like the ones we have utilized in our demo case promise to yield practical advantage compared to purely classical methods [7]. Therefore, there is also the question of how to account for hybrid algorithms in the context of MDE. Which conceptual extensions in modeling languages may be necessary for the design and efficient implementation of hybrid algorithms? Once this conceptual questions and challenges are solved, a next step is the design of according code generators for hybrid quantum/classical algorithms.

**RQ2: Intelligent Code Generators:** The mentioned code generators require the existence of idioms and patterns for QC in order to produce effective and efficient code. These include, beside others, standardized ways to represent certain problems and their solutions as well as repositories of quantum/hybrid algorithms and patterns with dedicated services, e.g., for automatic algorithm/pattern selection. In this context, a platform that serves similar purposes has been proposed by Leymann et al. [8]. This raises the question how such automatized processes may be modeled or reused in a code generator, e.g., how a NISQ-Analyzer for automated backend and parameter selection [17] may be utilized in MDE.

**RQ3: Deployment and Integration via Cloud-based Quantum Execution:** Currently, QC can be accessed via Cloud services. However, such Cloud-based Quantum execution requires dedicated deployment and integration mechanisms—especially for hybrid algorithms. Such mechanism have to account for the different machines which can be exploited and current modeling languages for Cloud computing [18] have to be extended to utilize QC.

**RQ4: Executable DSMLs:** The challenge of testing and debugging an executable DSML is important. The debugging of translational DSMLs is a major challenge even for classical computers. In the case of QC it is even more complicated to map an error from the hardware level back to the DSML due to the problem of measuring superposition states. Similar issues concerning such an estimation of superposition states have been raised by Ali & Yue [3] in the context of quantum model-based testing. On the hardware level, the different vendors of QC via Cloud services offer different tools for problem inspection. Therefore, again the question of how to perform the abstraction step to appropriate DSMLs has to be asked.

**RQ5: Model-Driven Evaluation of Quantum Technologies:** MDE may be utilized for the evaluation of existing QC technologies and the conduction of various experiments in an efficient manner. Different programs with varying input data may be generated automatically from models. Therefore, the modeling environment may serve to run domain-specific studies on QC technologies. As a precondition, the value of QC for the various DSMLs must be evaluated. Furthermore, it has to be assessed whether these DSMLs, e.g., in technical, social, scientific, or economic domains, would require adaptations in order to be feasible for QC representations.

**RQ6: Quantum Computing for solving MDE tasks:** The computation power of QC may help to solve hard problems in MDE. Examples include tasks such as model transformations, model-based optimization (as demonstrated in this paper), and model simulations. This may require extensions of meta-modeling languages and model transformations to be compatible with QC (cf. also RQ1).

**Next Steps.** For further traveling the outlined research roadmap, we provide our demonstration case with all artefacts as an open source repository [19] and invite the community to explore together the different RQs.

**ACKNOWLEDGMENTS**

Financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development and by the Austrian Science Fund (P 30525-N31) is gratefully acknowledged.

**DATA AVAILABILITY**

We share all data via the following URL: http://doi.org/10.5281/zenodo.4593888

**REFERENCES**


