2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)

Pattern-based Generation and Adaptation of Quantum Workflows

Martin Beisel, Johanna Barzen, Frank Leymann, Lavinia Stiliadou, Daniel Vietz, Benjamin Weder
University of Stuttgart, Institute of Architecture of Application Systems, Universitätsstrasse 38, 70569 Stuttgart, Germany
{*firstname.lastname*}@*iaas.uni-stuttgart.de*

***Abstract*—Building quantum applications requires deep knowledge of quantum computing and software engineering. Hence, an abstraction layer reducing the complexity for non-experts is needed. Patterns are an established concept for the abstract description of proven solutions to recurring problems. Therefore, the quantum computing patterns, a pattern language for the quantum computing domain, can be used to define the building blocks and the structure of hybrid quantum applications. Furthermore, concrete software artifacts can be associated with patterns to solve the corresponding problem. However, these software artifacts are usually heterogeneous, e.g., using different data formats. Quantum workflows enable a robust and scalable orchestration of these heterogeneous software artifacts. However, manually modeling and configuring such quantum workflows is a complex, error-prone, and time-consuming task. To overcome this issue, we present an approach that automates the generation and adaptation of quantum workflows using the quantum computing patterns. We provide an architecture realizing our approach, a corresponding prototype, as well as an evaluation comprising different use cases, a runtime comparison, and a user study.**

***Index Terms*—Quantum Computing, Workflows, Patterns**

I. Introduction

The rapid developments in the quantum computing domain promise computational advantages for real-world use cases in the near future [1, 2]. Since quantum devices are unsuitable for many traditional tasks, e.g., user interactions, they are typically used as co-processors in hybrid quantum-classical applications [3]. Further, quantum algorithms often require classical pre- and post-processing, e.g., to prepare data for the quantum device [4, 5]. Thus, the corresponding quantum and classical programs implementing these functionalities must be orchestrated, i.e., their control and data flow must be defined [6].

An established concept for orchestrating heterogeneous programs in various domains is workflows [7, 8]. By integrating the different programs of a hybrid quantum application using workflows, they can benefit from their advantages, e.g., robustness and scalability [6]. However, modeling such quantum workflows is complex, time-consuming, and requires knowledge of quantum algorithms, mathematics, and software engineering. To facilitate the modeling of quantum workflows, the *Quantum Modeling Extension (QuantME)* has been introduced [9]. QuantME provides dedicated workflow modeling constructs for various recurring quantum tasks and defines corresponding configuration properties to customize these tasks for a certain use case. While QuantME eases modeling and configuring quantum tasks for experts, the wide range of options is complex for users without deep quantum knowledge.

To enable users without knowledge of quantum computing and workflow technologies to benefit from the advantages of quantum workflows, another layer abstracting further technical details may be helpful. Patterns are a well-known concept to capture proven solutions for recurring problems in an abstract and easily digestible manner [10]. While patterns originate from the architecture domain, they have been successfully applied in different domains, particularly in software engineering, e.g., enterprise integration patterns [11] and software design patterns [12]. To summarize important concepts from the quantum computing domain and educate developers, a domain-specific pattern language was introduced [13]. By combining different quantum computing patterns, users can abstractly describe the building blocks and the structure of hybrid quantum applications. However, executing these quantum applications requires the availability of the programs and services realizing the functionalities of the utilized patterns.

Solution languages enable associating patterns with programs that solve the corresponding problems [14]. These programs are often heterogeneous, using differing programming languages and data formats. To benefit from their advantages, quantum workflows can be used to orchestrate the programs [6]. However, manually modeling these quantum workflows weakens the benefits of abstract, pattern-based modeling.

In this paper, we present an approach for the pattern-based modeling and execution of hybrid quantum applications. It enables selecting a set of quantum computing patterns and automatically generates a quantum workflow orchestrating their corresponding solutions. Furthermore, patterns can be attached to modeling constructs of quantum workflows, which are automatically adapted to integrate the functionality realized by the solutions of the patterns. This also enables incorporating novel quantum computing techniques, such as circuit cutting [15] or warm-starting [16], into existing quantum applications. To realize our approach, we present a system architecture and a prototype implementing it. We evaluate our approach (i) by a case study comprising three typical use cases from the quantum computing domain, (ii) a runtime evaluation of the automated steps of our approach based on different application scenarios, and (iii) a comparison of the time different user groups need when using our approach in contrast to manually modeling quantum workflows. Finally, we (iv) conducted semi-structured interviews with partners from academia and industry, assessing the usability of the approach as well as the corresponding prototype, and discussing possible improvements.

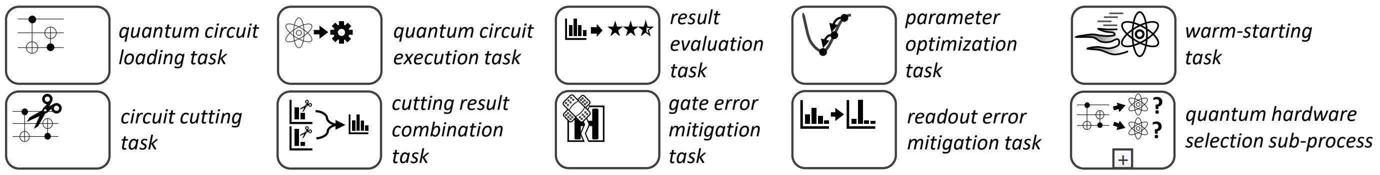
quantum circuit loading task
quantum circuit execution task
result evaluation task
parameter optimization task
warm-starting task
circuit cutting task
cutting result combination task
gate error mitigation task
readout error mitigation task
quantum hardware selection sub-process

Fig. 1. Overview of the QuantME modeling constructs and their graphical notation

The remainder of this paper is structured as follows: Section II discusses fundamentals and presents the problem statement of this work. In Section III, the approach for the pattern-based generation and adaptation of quantum workflows is introduced. Section IV showcases the system architecture realizing it and the corresponding prototypical implementation. Section V evaluates our approach, and Section VI discusses potential limitations and threats to validity. Finally, Section VII presents related work, and Section VIII concludes the paper.

II. Fundamentals & Problem Statement

This section provides the fundamentals about quantum workflows, pattern and solution languages, and the quantum computing patterns. Furthermore, the problems underlying this work and the corresponding research question are discussed.

A. Quantum Workflows & QuantME

Workflow technologies enable the modeling of processes comprising various activities, e.g., service invocations or user tasks, as well as the control and data flow between them [7]. Thereby, standardized workflow languages, such as BPMN [17] or BPEL [18], enable defining interoperable workflows. These workflows can be executed using mature and reliable workflow engines [7]. Workflow engines provide various benefits, such as robustness and scalability [19, 20]. Therefore, they are a well-established technology for orchestrating heterogeneous applications in different domains, such as business process management [8], e-science [21] or quantum computing [9].

To facilitate the modeling and execution of quantum workflows, the *Quantum Modeling Extension (QuantME)* has been introduced [9, 22]. QuantME provides custom-tailored modeling constructs for commonly occurring tasks in the quantum computing domain. An overview of the QuantME modeling constructs relevant to this work is shown in Figure 1. Each QuantME modeling construct defines type-specific properties, easing their configuration for the use case at hand. The *circuit loading task* enables loading a quantum circuit to solve a certain problem into a workflow. The *quantum circuit execution task* provides configuration properties for defining which quantum hardware provider and which specific quantum device should be utilized for executing a quantum circuit. To evaluate the quality of quantum circuit execution results, the *result evaluation task* can be utilized. Different algorithms, e.g., variational quantum algorithms, require the classical optimization of parameters within a quantum circuit [23]. The *parameter optimization task* provides configuration properties for setting up this optimization routine. Various warm-starting techniques have been introduced to improve the efficiency of quantum algorithms, e.g., by pre-computing good initial parameter values [16]. To incorporate these techniques into quantum workflows, the *warm-starting task* has been introduced. Circuit cutting techniques enable splitting a quantum circuit into smaller sub-circuits that can be executed by quantum devices that otherwise would have an insufficient number of qubits [24]. To integrate these techniques, the *circuit cutting task* enables splitting circuits and the *cutting result combination task* provides functionalities to combine the execution results of the corresponding sub-circuits. Current quantum devices are prone to different types of errors, e.g., readout errors and gate errors. To reduce the impact of these errors on the circuit execution results, various error-handling techniques have been introduced [25]. Error-handling techniques mitigating the impact of gate and readout errors can be integrated using the *gate error mitigation task* and *readout error mitigation task*, respectively. Finally, the *quantum hardware selection sub-process* enables the automated selection of a suitable quantum device for the circuit that shall be executed.

To ensure the executability of QuantME workflows using existing workflow engines, the contained QuantME modeling constructs are replaced by native modeling constructs of the used workflow language in an automated transformation method [9]. This transformation is based on reusable workflow fragments, which are defined by domain experts. For this, so-called *QuantME Replacement Models (QRMs)* are utilized, consisting of two parts: (i) First, a *detector* specifies which QuantME modeling constructs the QRM can replace. This detector includes a single task of a certain QuantME task type, as well as corresponding configuration properties. For example, a detector defining a parameter optimization task configured with a specific optimizer, e.g., SPSA [26] or Adam [27], indicates that the QRM supports the corresponding optimizers. (ii) Second, a *replacement fragment* comprises the workflow fragment implementing the specified functionality. This workflow fragment is inserted during the transformation to replace the identified QuantME modeling construct. For example, to replace a parameter optimization task identified through the previously mentioned detector, a fragment invoking a service providing the required optimization technique can be used.

B. Pattern & Solution Languages

Patterns are a concept for describing proven solutions to recurring problems in a structured manner [10]. Each pattern follows a well-defined format, comprising a problem statement, its context, and forces, i.e., requirements or constraints restrict-

TABLE I
OVERVIEW OF THE PATTERNS AND THEIR CATEGORIZATION

Pattern Category	**Patterns**
Algorithm	Alternating Operator Ansatz (AOA), Quantum Approximate Optimization Algorithm (QAOA), Quantum Kernel Estimator (QKE), Quantum KMeans, Quantum Phase Estimation (QPE), Quantum Support Vector Machine (QSVM), Variational Quantum Eigensolver (VQE)
Behavioral	Orchestrated Execution, Pre-deployed Execution, Prioritized Execution, Quantum Hardware Selection
Enhancement	Biased Initial State, Chained Optimization, Circuit Cutting, Error Correction, Gate Error Mitigation, Pre-Trained Feature Extractor, Readout Error Mitigation, Variational Parameter Transfer

ing the possible solutions [12]. Further, an approach for solving the problem is presented in an abstract manner with a corresponding sketch, and the implications of the solution are discussed. To showcase the practical applicability of the pattern, a known uses section describes real-world use cases where the pattern was successfully applied. Finally, related patterns are connected using semantic relations. The interconnections between all patterns of a domain are used to form a so-called *pattern language* [10]. Depending on the pattern language, additional sections may be added to the pattern descriptions, e.g., a section about examples of how the pattern can be applied.

To facilitate the application of abstract solutions provided by pattern languages, Falkenthal et al. [14] introduce the concept of *solution languages*. Solution languages comprise various *concrete solutions*, i.e., implementations that realize the solution to solve a pattern's problem for a specific scenario or programming language. Concrete solutions are directly linked to the patterns they realize and are connected to related concrete solutions in the solution language to enable their combination.

C. Quantum Computing Patterns

To support quantum software engineers in developing quantum applications, the quantum computing pattern language has been introduced [13]. Since its introduction, the pattern language has been continuously extended to cover different aspects of quantum applications, quantum algorithms, and their building blocks. To generate and adapt quantum workflows utilizing patterns, we group the quantum computing patterns into three categories: (i) *Algorithm patterns*, such as the *Quantum Approximate Optimization Algorithm (QAOA)* [28], explain the concepts and application scenarios of established quantum algorithms. (ii) *Behavioral patterns*, e.g., orchestrated execution or pre-deployed execution [29], discuss different concepts for executing quantum applications. (iii) *Enhancement patterns*, such as the quantum error handling [25] and warm-starting [30] patterns, describe concepts for improving the result quality. An overview of all patterns and their categorization is shown in Table I. An in-depth description of each pattern as well as the semantic relations between them can be found on the PlanQK Platform [31], a public community platform for sharing knowledge about quantum computing.

D. Problem Statement & Research Question

As previously discussed, workflows enable the orchestration of hybrid quantum algorithms as well as their integration into existing applications. However, the modeling of quantum workflows is complicated by the complexity of quantum computing [9]. To reduce this complexity and ease the understanding of crucial quantum computing concepts, patterns can serve as an additional abstraction layer [13]. However, to facilitate the application of the quantum computing patterns, a corresponding solution language is required. Thus, we introduce a draft of the solution language comprising various implementations. Implementations in the quantum computing domain are typically realized using script-based languages, such as Python and JavaScript [32]. However, script-based languages do not provide the sophisticated functionalities of mature workflow engines [7]. Manually migrating these implementations to workflow-based solutions is complex, error-prone, and time-consuming. This leads us to our first Research Question (RQ):

RQ1: *"How to automatically generate quantum workflows based on a set of selected quantum computing patterns?"*

To address our research question, we divide it into two questions focusing on the main problems. The basic functionality for a set of selected quantum computing patterns is realized by the algorithm patterns. In contrast, the behavioral and enhancement patterns define how they are executed or improved. Therefore, to generate a quantum workflow, first, the concrete solutions of the algorithm patterns must be orchestrated.

"How can concrete solutions associated with algorithm patterns be orchestrated as part of a quantum workflow?"

Although the generated quantum workflows are already executable, incorporating behavioral and enhancement patterns enables use-case-specific optimizations, which provide various benefits, such as improved accuracy and increased efficiency.

"How to use behavioral and enhancement patterns to enrich quantum workflows with additional functionalities?"

The availability of an approach to automatically generate quantum workflows does not imply that it is more suitable for developing quantum applications solving real-world problems than existing approaches that generate quantum applications or a pure manual development. As using a new approach always adds additional complexity and learning effort, it must be verified that it is easy to use and learn even without deep knowledge of quantum computing and workflow technologies. Furthermore, its advantages, e.g., reducing the development time, must be evaluated. Thus, our second RQ is as follows:

RQ2: *"How much can the introduced approach decrease the complexity and time required when developing quantum applications compared to existing approaches?"*

III. PATTERN-BASED QUANTUM WORKFLOWS

In the following, we discuss our approach for the generation and adaptation of quantum workflows based on the quantum computing pattern language, which is depicted in Figure 2.

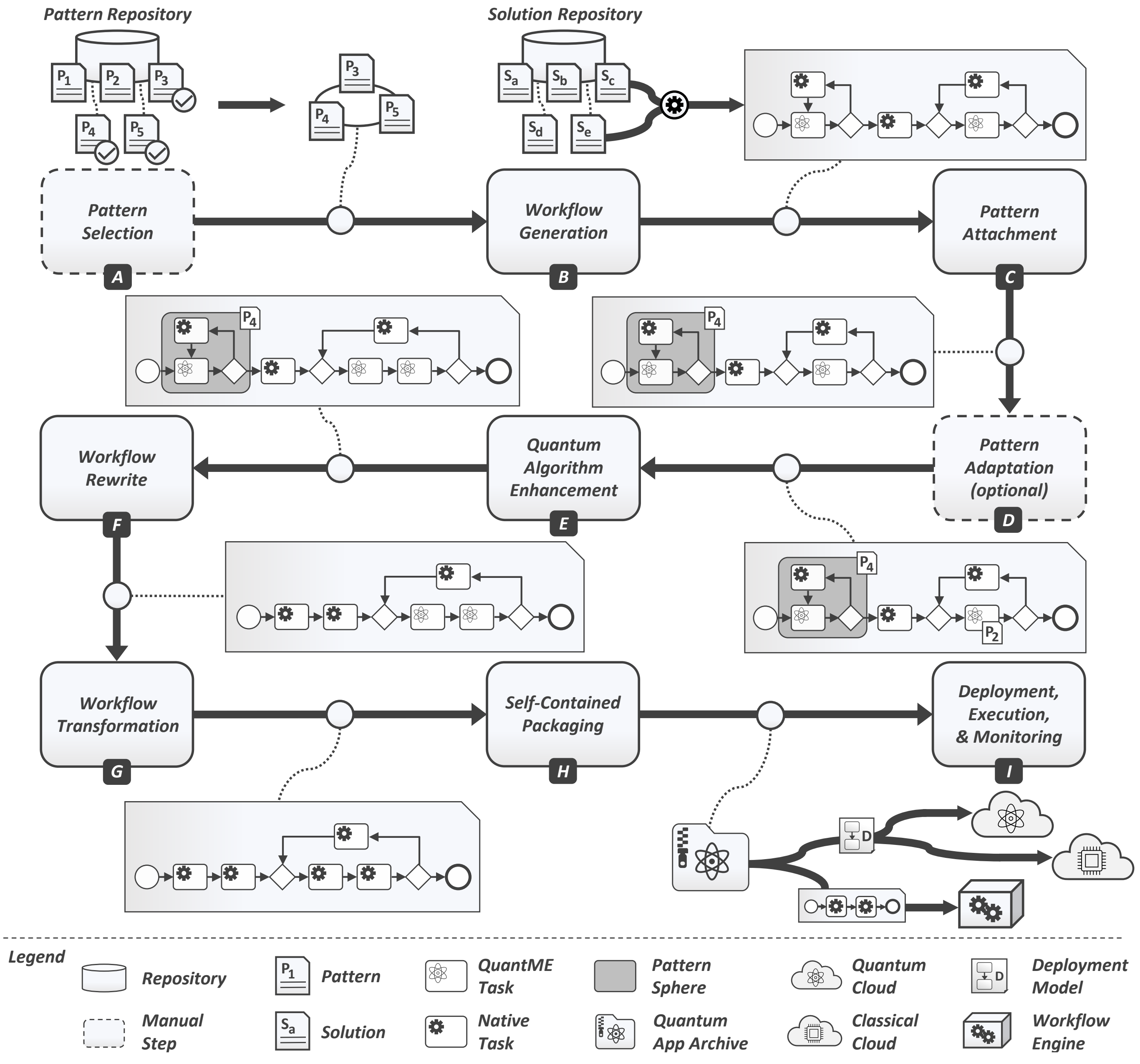

Fig. 2. Overview of the approach for the pattern-based generation and adaptation of quantum workflows

A. Pattern Selection

In the pattern selection step, the user chooses one or multiple algorithm patterns from the pattern repository and defines their partial order. For each selected algorithm pattern, corresponding behavioral and enhancement patterns can be manually picked to customize algorithms. While they improve the execution, the usage of these patterns typically includes additional processing, increasing execution time and costs. Therefore, these patterns should be selected on a case-by-case basis.

While pattern languages foster combining multiple related patterns to solve a larger problem, different patterns also inherently forbid the selection of other incompatible patterns within the pattern language [10]. These incompatibilities are specified by the pattern authors in the related patterns sections utilizing a connection of type *canNotBeCombined*. Incompatibilities exist (i) between different enhancement patterns, e.g., the gate error mitigation and the error correction pattern exclude each other as no errors need to be mitigated if all errors are canceled out by the error correction technique, and (ii) between algorithm patterns and specific enhancement patterns, e.g., the chained optimization pattern can only be used in combination with algorithm patterns derived from the VQA pattern, such as QAOA and VQE. All of these pattern relationships are stored in the pattern repository. To facilitate the manual pattern selection process, tooling support can display these incompatibilities, preventing users from selecting incompatible patterns.

B. Workflow Generation

In the second step, the quantum workflow is generated based on the selected algorithm patterns. Thereby, a solution repository is searched for suitable implementations of the respective patterns. As previously discussed, algorithm implementations are commonly available as monolithic scripts. However, orchestrating a quantum algorithm using a single script task that directly executes the given implementation would (i) revoke the benefits of workflow engines, and (ii) hinder the customization of quantum algorithms by means of behavioral and enhancement patterns as they may add additional steps within the algorithm. Therefore, the scripts implementing the quantum algorithms are split into their individual building blocks [32]. For example, the circuit generation and circuit execution can be separated into distinct QuantME tasks. If the concrete solution of an algorithm pattern is available as a workflow it can be directly integrated into the overall workflow. This can, e.g., be achieved using a call activity in BPMN [17].

C. Pattern Attachment

Next, the selected behavioral and enhancement patterns are integrated into the quantum workflow. Enhancement patterns are directly attached to suitable QuantME modeling constructs orchestrating the corresponding quantum algorithm. For example, a readout error mitigation pattern is attached to the quantum circuit execution task within the algorithm, as it operates on the execution results. In contrast to enhancement patterns, behavioral patterns are not applied to single tasks. Rather, they affect the behavior of a set of tasks, to which we refer to as a *pattern sphere*. Therefore, for each algorithm pattern with at least one linked behavioral pattern, an initial pattern sphere, comprising all corresponding tasks, is created. Furthermore, the respective behavioral patterns are attached to the pattern sphere. For example, in Figure 2, P_4 is a behavioral pattern that was selected together with the first algorithm pattern in the first step. Hence, it is attached to a pattern sphere comprising all tasks orchestrating the first quantum algorithm.

D. Pattern Adaptation

The fourth step of our approach is optional and allows users to manually attach and remove enhancement and behavioral patterns. Therefore, patterns that were not selected in the first step of our approach can be manually added to a quantum workflow. Whereas enhancement patterns are attached to tasks, behavioral patterns are attached to the initial pattern spheres created in the previous steps. Furthermore, users can create new pattern spheres, delete existing ones, and adapt them by adding or removing tasks. To prevent users from adding incompatible patterns to the quantum workflow, an automated validation is required. For enhancement patterns, the suitability of the task type as well as the compatibility of the already attached patterns must be checked. For example, a circuit cutting pattern must be attached to a quantum circuit execution task, as it changes the semantics of the quantum circuit execution and hence can not be attached to other types of tasks, such as a parameter optimization or result evaluation task.

E. Quantum Algorithm Enhancement

In the fifth step of our approach, all enhancement patterns attached to the workflow are processed. Since QuantME was specifically designed with the quantum computing patterns in mind, there are dedicated tasks for each enhancement pattern. Hence, for all attached enhancement patterns, a suitable QuantME task is injected into the workflow. The placement of QuantME tasks is determined based on pattern-specific injection rules. For example, the implementation of a warm-starting pattern is orchestrated using a warm-starting task, which must be placed in front of the quantum circuit loading task the pattern is attached to. The approach for transforming the injected QuantME tasks using the concrete solutions of the corresponding patterns is discussed in Section III-G.

F. Workflow Rewrite

After all enhancement patterns have been applied, the behavioral patterns are processed to change the semantics of all tasks contained in the corresponding pattern sphere. As pattern spheres comprise a set of tasks, sophisticated approaches for rewriting quantum workflows are required. For example, the prioritized execution pattern [29], which describes how waiting times of quantum executions can be reduced, is realized in the workflow by adding tasks for setting up a session providing prioritized access to a quantum device. However, it also requires adapting the circuit execution to use the created session. Moreover, behavioral patterns can be combined, e.g., the pre-deployed execution and the prioritized execution pattern lead to a workflow rewrite for a hybrid runtime. Hybrid runtimes reduce the latency, by deploying classical code close to the used quantum device [33]. Hence, workflows that alternate between executions on classical and quantum devices benefit from such hybrid runtimes. A typical example are *Variational Quantum Algorithms (VQAs)* [5], which iteratively improve the execution result by alternating between the execution of parameterized quantum circuits and the classical optimization of these parameters. To execute the respective parts of the workflow in a hybrid runtime, they are combined into a single artifact that is pre-deployed in the hybrid runtime and then invoked during workflow runtime [33].

G. Workflow Transformation

Next, all QuantME modeling constructs are transformed into native workflow modeling constructs of the used workflow language. Thereby, a repository containing suitable workflow fragments, implementing the functionality of the QuantME modeling constructs, is utilized [9]. Each workflow fragment defines a detector that specifies which type of QuantME modeling construct it supports. Moreover, the applicability of the workflow fragment can be further restricted by using a set of type-specific configuration properties. For example, the detector of type quantum circuit execution task provides properties to define the set of supported quantum providers.

However, this transformation requires the availability of suitable workflow fragments for each QuantME modeling construct within the workflow. While the solution repository

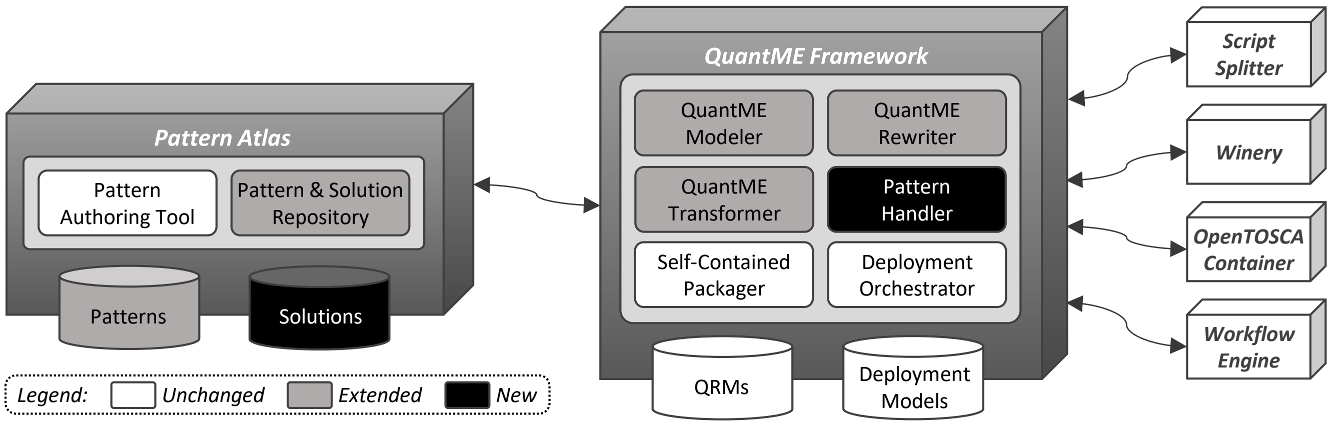

Fig. 3. Overview of the system architecture for the pattern-based generation and adaptation of quantum workflows

contains a wide range of available implementations for various patterns, they can not be directly used to replace QuantME tasks as they lack (i) a detector, and (ii) are not always realized as workflows. To generate workflow fragments and corresponding detectors based on concrete solutions, we apply a semi-automatic approach: When adding a concrete solution to the solution repository, the QuantME task type can be automatically derived based on the respective pattern. However, as automatically retrieving suitable configuration properties for the detector is not always possible, users can manually define them. In addition to concrete solutions provided as workflows, our approach enables the utilization of scripts and services. Script-based implementations are transformed into workflows as described in step B for algorithm patterns. Services are provided via a *service registry* [34], enabling the generation of suitable requests via the API specifications of the corresponding services that are then invoked by the workflow.

H. Self-Contained Packaging

Since packaging applications is crucial to share, distribute, and sell them, in step H, all required software artifacts of the quantum workflow are bundled in a self-contained quantum application archive [6]. This includes the packaging of the workflow, as well as all code required to orchestrate the application, e.g., the implementations of the different patterns.

I. Deployment, Execution, & Monitoring

In the last step of our approach, the previously packaged artifacts are deployed in the execution environment of the workflow [6]. However, the manual deployment of implementations and their required dependencies is complex, time-consuming, and error-prone. To tackle this issue, deployment technologies automating this process were introduced [35]. We facilitate the deployment of the required implementations by automatically generating deployment models for them [32]. After executing all deployment models utilizing deployment systems, such as Kubernetes or Terraform, the endpoints of the implementations are bound to the respective tasks of the workflow. Finally, the quantum workflow is uploaded to the workflow engine and can be instantiated by passing the required input parameters, e.g., the token to access a quantum device or the problem instance to solve. During execution, the user can monitor the progress of the workflow. Further, all data produced during execution is stored in the *audit trail* [36], enabling an in-depth analysis of the workflow executions [6].

IV. Architecture & Prototype

In this section, we introduce a system architecture for the presented approach and showcase the prototype realizing it.

A. System Architecture

An overview of the system architecture supporting our approach is shown in Figure 3. The system architecture comprises six components that handle the management of patterns, the modeling and execution of workflows, the splitting of scripts, and the automated deployment of implementations.

Pattern Atlas [37] is a tool for authoring and managing patterns and pattern languages, as well as corresponding concrete solutions. The *pattern authoring tool* supports the collaborative pattern authoring process with a pattern editor. Two databases are used to store *patterns* and corresponding *solutions*, which are managed by the *pattern & solution repository*. As the Pattern Atlas only provides tooling for patterns and not corresponding solutions, the solution repository is newly added. This also requires adding the related management functionality in the pattern & solution repository.

The *QuantME Framework* [38] enables modeling workflows utilizing BPMN and QuantME modeling constructs. The graphical modeling is performed using the *QuantME modeler*, which has been extended to provide modeling capabilities for the quantum computing patterns. This comprises adding a selection modal enabling choosing the algorithm and corresponding behavioral and enhancement patterns in step A. Furthermore, it enables to drag-and-drop patterns to attach them to activities of existing quantum workflows. The *QuantME rewriter* enables the application of behavioral patterns to the workflow. It is plugin-based and uses existing methods, such as the rewrite for hybrid runtimes [33]. Moreover, new methods, e.g., to rewrite the workflow for prioritized access via sessions. The *QuantME transformer* ensures the portability and interoperability of workflows containing QuantME modeling constructs and enables their transformation into native BPMN modeling constructs using reusable workflow fragments stored in the *QuantME Replacement Model (QRM)* repository. Additionally, newly added functionality orchestrates the generation of suitable QRMs when adding new concrete solutions to the repository. The new *pattern handler* manages the pattern attachment and facilitates the manual pattern adaptation by analyzing the validity of the user inputs, e.g., if the selected enhancement patterns are compatible. Furthermore, it enables injecting suitable QuantME tasks for the attached enhancement patterns. Finally, the pattern handler also invokes the script splitter to generate workflow fragments on the basis of script-based concrete solutions, e.g., realizing an algorithm pattern or a solution for an enhancement pattern [32]. To enable distributing quantum applications, the *self-contained packager* bundles the quantum workflow and all other required artifacts in a self-contained quantum application archive. The *deployment orchestrator* handles all deployment-related functionalities: (i) The generation of deployment models for concrete solutions using *Winery* [39], a graphical modeling tool based on the TOSCA standard [40], (ii) the deployment of all required services utilizing the *OpenTOSCA Container* [41], a TOSCA-compliant deployment system, and (iii) the upload of the executable workflow model to the *workflow engine*.

B. Prototypical Implementation

The prototypical implementation of our system architecture is publicly available on GitHub [42]. The Pattern Atlas is realized using Java and TypeScript and utilizes a PostgreSQL database for storing patterns and concrete solutions. Quantum computing patterns are included in the Pattern Atlas once they are peer-reviewed by the scientific community. Anyone can contribute concrete solutions through a GitHub pull request, which will be reviewed and validated by domain experts. The QuantME Framework is a web-based component that is written in JavaScript and utilizes the bpmn-js framework, providing an extensible BPMN modeler. It was extended using multiple plugins, implementing the QuantME and pattern functionalities.

V. Evaluation

In this section, we evaluate the practical feasibility of our approach and analyze how much it supports developers in implementing quantum workflows. For this, we first analyze its applicability and suitability for three exemplary use cases from the quantum computing domain. Second, we evaluate the runtime of all automated steps using our prototype based on different application scenarios to identify possible bottlenecks. Subsequently, we perform a user study to compare the time user groups with different backgrounds require when realizing one of the use cases with our approach compared to a manual modeling approach. Finally, we conduct semi-structured interviews with the participants of the user study to assess the usability of our approach and the corresponding prototype and discuss problems and possible improvements.

A. Case Study

To validate the practical feasibility of our approach, we apply it to three exemplary use cases from the quantum computing domain. Thereby, required patterns are selected and the quantum workflows realizing the use cases are automatically generated. A demonstration video showcasing all steps of our approach can be found on YouTube [43]. Furthermore, the implementations of all use cases are available on GitHub [42].

In the first use case, the Maximum Cut problem is solved by employing QAOA, a typical VQA that promises to solve optimization problems efficiently [28]. Figure 4 gives an overview of the different steps when applying our approach to this use case. First, the required algorithm pattern is selected, i.e., the QAOA pattern, as shown on the top left. Further, the execution of the quantum algorithm should be improved using three enhancement patterns: (i) Biased initial state, (ii) readout error mitigation, and (iii) circuit cutting. Biased initial state is a warm-starting technique that computes an approximation for the problem instance to solve, e.g., to compute an approximation for the Maximum Cut problem the Goemans-Williamson algorithm [44] can be used [30]. This approximation is encoded into the quantum circuit to ease finding the optimal solution with the quantum computation [16]. The readout error mitigation pattern is intended to add functionality reducing the impact of errors due to decoherence during the measurement operations on a quantum device [25]. Finally, the circuit cutting pattern is selected to reduce the size of the executed quantum circuits, and thus, the noise in the execution results [15]. In addition to the enhancement patterns, the quantum hardware selection pattern was chosen as a behavioral pattern. This pattern enables the automated selection of a suitable quantum device for the execution of quantum circuits based on their properties as well as the current characteristics of the available quantum devices. After the pattern selection, the quantum workflow implementing the selected algorithm pattern is generated. In this use case, a quantum workflow orchestrating the QAOA algorithm is already present in the solution repository. Thus, this solution is loaded and the behavioral and enhancement patterns are attached to the corresponding sub-process, as depicted in the middle of Figure 4. This sub-process consists of four tasks. First, the quantum circuit realizing QAOA is generated. As this QAOA circuit is parameterized, an initial parameterization for the quantum circuit is generated based on the input data. Subsequently, within an optimization loop, the quantum circuit is executed, the execution results are evaluated, and the parameters are classically optimized based on the evaluated results until the optimization converges and no better parameterization can be found. In the last workflow shown at the bottom of Figure 4, the behavioral and enhancement patterns are incorporated. A warm-starting task is added directly after the start event, which realizes the attached biased initial state pattern by computing an initial approximation for the Maximum Cut problem which is encoded into the quantum circuit as an initial state. To realize the readout error mitigation pattern a readout error mitigation task is injected after the quantum

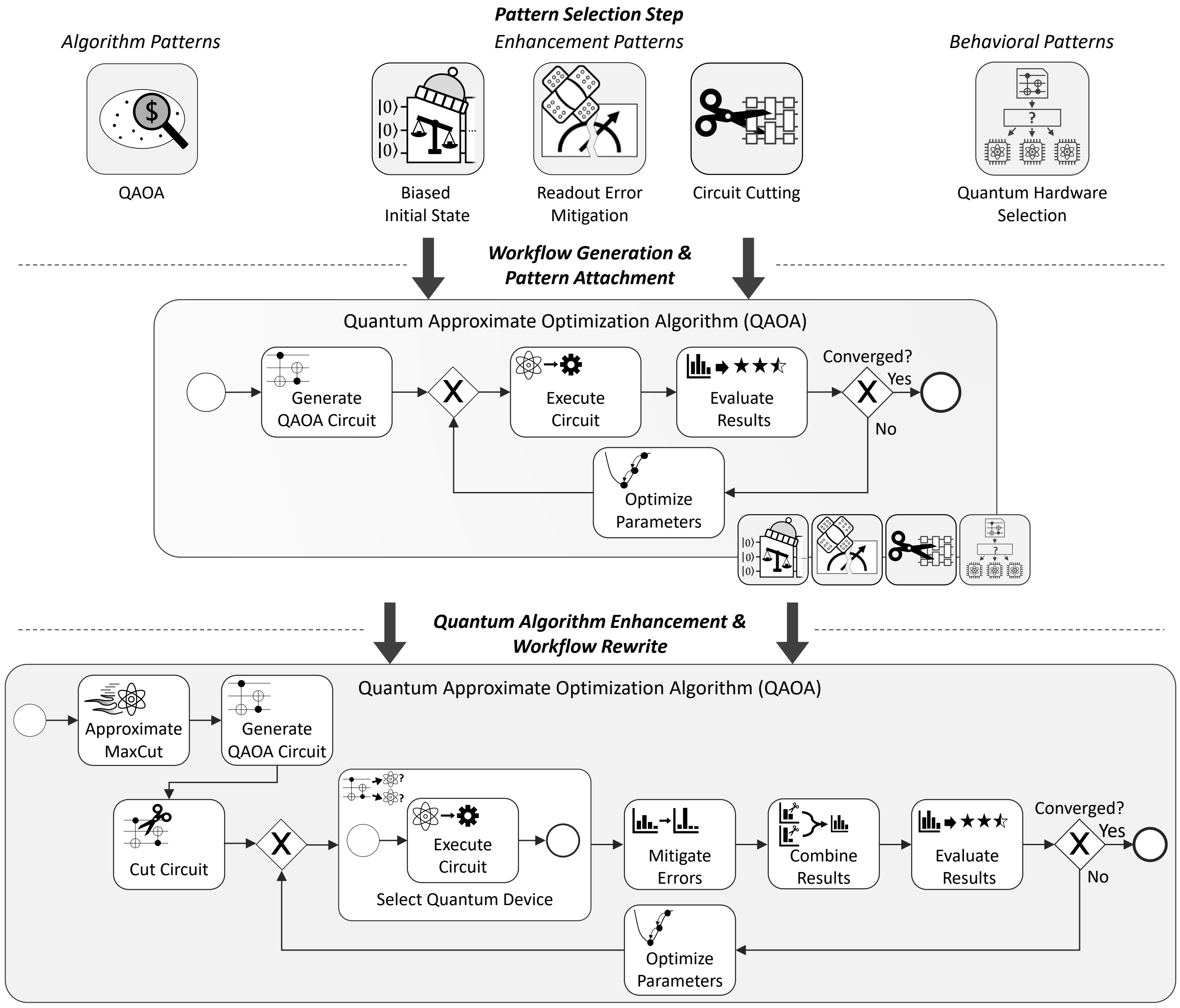

Fig. 4. Overview of the different steps of our approach for the QAOA use case

circuit execution task to improve the execution results. The circuit cutting pattern requires the insertion of two tasks: First, a circuit cutting task after generating the quantum circuit and before entering the optimization loop, and second, a cutting result combination task, which combines the quantum circuit execution results after each execution in the loop. Finally, to realize the quantum hardware selection pattern, the quantum circuit execution task is wrapped into a quantum hardware selection sub-process that enables the automated selection of a suitable quantum device for the given quantum circuit.

The second use case is from the machine learning domain and utilizes the quantum k-means algorithm [45] to cluster input data and then trains a quantum support vector machine [46] based on the resulting clusters. The trained quantum support vector machine can be utilized to classify further input data. In contrast to the first use case, the quantum algorithm is available as a script and, therefore, must be transformed into a workflow using the script splitter in the workflow generation phase. Furthermore, the *Quantum Machine Learning (QML)* use case incorporates two behavioral patterns, namely the pre-deployed execution pattern and prioritized execution pattern, leading to the execution in a hybrid runtime environment [33].

In the last use case, eigenvalues of matrices are computed utilizing a *Variatonal Quantum Eigensolver (VQE)* [47]. Efficiently computing eigenvalues is crucial for various application areas, e.g., drug discovery and material design [48]. Similar to the second use case the algorithm is available as a script that must be transformed into a workflow utilizing the script splitter. To facilitate the optimization of the parameters of the VQE circuits, the variational parameter transfer pattern is applied. This pattern aims to pre-compute suitable initial circuit parameters that can be utilized as starting points for the optimization.

TABLE II

CHARACTERISTICS OF ALL USE CASES AND THE CORRESPONDING RUNTIMES OF ALL AUTOMATED STEPS UP TO THE SELF-CONTAINED PACKAGING

Use Case	Number of Patterns			#Scripts to Split	Number of Activities		#Depl. Models	Median Runtime in Seconds					
	Alg.	Enh.	Beh.		After Step B	After Step G		Step B	Step C	Step E	Step F	Step G	Step H
QAOA	1	3	1	0	6	12	4	0.69	0.06	0.31	0.01	11.71	17.06
QML	2	0	4	2	14	6	6	448.42	0.11	0.01	460.82	11.71	49.04
VQE	1	1	0	2	10	17	14	259.23	0.03	0.08	0.01	300.04	90.65

B. Runtime Evaluation

In the following, we discuss the runtime of our approach for the three presented use cases, as summarized in Table II. The use cases are characterized by different properties, influencing the runtime of the method steps: First, the number of used patterns, i.e., the algorithm patterns, the enhancement patterns, and the behavioral patterns. The number of algorithm patterns corresponds to the number of sub-workflows that have to be combined and directly impacts the time for the workflow generation in step B. Moreover, it also affects the number of activities of the generated workflow, and a higher number potentially increases the execution time of all remaining steps. In contrast, the number of enhancement patterns only influences the placement of the patterns in step C, as well as the quantum algorithm enhancement in step E. The behavioral patterns have to be placed in step C and are processed in step F, thus, their number impacts the execution times of these steps. The number of scripts that must be split during workflow generation in step B and workflow transformation in step G strongly affects the execution time. The reason for this is the required code analysis and generation of services corresponding to the identified script parts. However, new script-based solutions can also be split in an offline pre-processing step when added to the solution repository. Furthermore, the number of activities can potentially increase the execution time of all steps. Therefore, the initial number of activities after executing step B, as well as the final activities after step G are listed in Table II. The number of deployment models directly impacts the time for bundling all required software artifacts of the quantum workflow in a self-contained archive, as all deployment models need to be downloaded from the deployment model repository.

On the right side of Table II, the runtimes of the different steps are shown. Thereby, the median based on ten measurements for each use case is calculated. All measurements were performed on a computer running Windows 11 with an AMD Ryzen 7 PRO 4750U and 32 GB of memory. The collected raw data is available on GitHub [49]. For the QAOA use case, a predefined workflow is loaded from the solution repository leading to a short workflow generation time in step B. In contrast, the QML use case includes two scripts that must be split in step B increasing the execution time. Furthermore, both algorithms for this use case are rewritten for a hybrid runtime. This also requires analyzing and changing code, thus, step F is the second step with a longer execution time. Similarly, the VQE use case requires splitting a script in step B and another script when transforming the workflow in step G. Hence, these steps dominate the overall execution time of the approach.

C. User Study

To evaluate how much our approach reduces the time users require to build quantum applications, we performed a user study utilizing the previously discussed QAOA use case. The task was to implement this use case (i) using the pattern-based approach and (ii) by manually modeling the quantum workflow. For the manual modeling of the quantum workflow, a guideline [50] was provided explaining how to use the QuantME modeling constructs to realize the use case. Overall, 20 participants with a background in computer science or physics were asked to participate in our user study. To achieve a diverse set of participants, we asked ten partners from academia and industry, respectively. Before tackling the given task, the users were asked to assign themselves to an expert or non-expert group based on their self-judgment about their experience in modeling workflows and their knowledge of quantum computing. Out of the 20 participants, 14 classified themselves as non-experts, while six assigned themselves to the expert group.

The required time to solve the task using both approaches is shown in Figure 5, and the anonymized raw data is available on GitHub [51]. On the left, the time required by the participants of the non-expert group is depicted. While the median time for manually modeling the quantum workflow was approximately 27 minutes, the pattern-based approach only required 4:24 minutes. Thus, a speed-up of factor 6 could be achieved by using our approach. The participants of the expert group finished the manual modeling in approximately 16 minutes, and thus, 11 minutes faster than the non-experts. The experts also achieved a speed-up of 1 minute using the pattern-based approach, however, the relative speed-up is smaller than for the manual modeling. Hence, the results show that although our approach provides a benefit in the development time for both user groups, it is even more beneficial for non-experts.

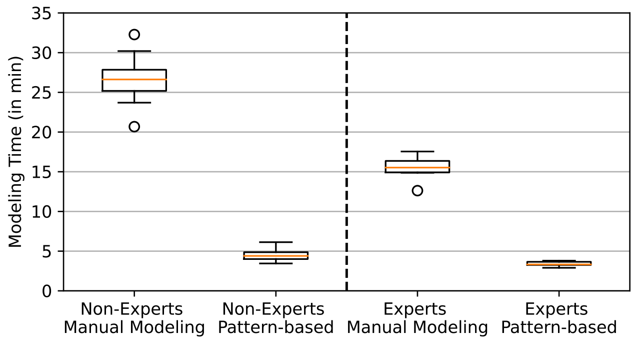

Fig. 5. Required time to solve the QAOA use case

D. *Interview-based Evaluation*

To evaluate the useability of our approach and the corresponding prototype, as well as to discuss potential limitations and improvements, we conducted semi-structured interviews with the participants of the user study from the previous section.

First, users were asked about the benefits and drawbacks they see when applying our approach compared to the manual modeling and other existing approaches they know. The users unanimously agreed that our approach facilitates the modeling of quantum workflows and reduces complexity. However, understanding all available patterns was an entry barrier for users who were not familiar with the quantum computing pattern language. Experts appreciated that they were able to apply the same custom configuration options as with the manual modeling while reducing the modeling effort. Furthermore, the reuse of existing software artifacts was mentioned as one of the main benefits of the method. Both user groups criticized that due to the automation, not all transformation steps were easily comprehensible. While users liked the wide variety of enhancement patterns, particularly quantum experts were missing some algorithms that they employ for their use cases.

In the second part of the interviews, we asked users for extensions or adaptations that could help us to improve our approach. To ease the selection of enhancement and behavioral patterns, users suggested including an automated recommendation system, which we plan to investigate as part of our future work. Additionally, users recommended certain domains to identify further algorithm patterns, e.g., quantum machine learning algorithms. As the quantum computing pattern language is still evolving, we expect further patterns to be identified increasing the general applicability of our approach.

VI. Discussion & Limitations

A common issue when transforming models using multiple steps is commutativity, as otherwise applying one transformation step can modify a part of the model, which is also required as input by another transformation step. Thus, the second transformation step can not be performed. This can lead to two problems: (i) Non-deterministic transformations reducing reproducibility and understandability, and (ii) deadends leading to incomplete transformations that require inefficient rollbacks of transformation steps. The quantum algorithm enhancement (step E) is commutative. For each enhancement pattern attached to a task or subprocess of a workflow, exactly one injection rule exists. These injection rules define how the QuantME tasks corresponding to the enhancement pattern are added to the workflow. In addition, they also specify the order in which the patterns have to be processed. For example, when a gate error mitigation pattern and a circuit cutting pattern are attached to a quantum circuit execution task, the circuit cutting pattern must be processed first, as the mitigation must modify the quantum circuits that are finally executed. However, the execution of the original circuit is substituted by the execution of the circuits resulting from circuit cutting. During workflow rewrite (step F), multiple behavioral patterns attached to a single pattern sphere are processed analogously to the previously discussed quantum algorithm enhancement step. To achieve commutativity between the processing of multiple pattern spheres, they must not overlap. Thus, rewriting a pattern sphere only applies local changes not affecting other pattern spheres. The workflow transformation (step G) replaces each QuantME task independently of the rest of the workflow. Therefore, transforming quantum workflows is commutative.

The main challenge when generating quantum workflows utilizing script-based implementations is to find suitable splitting points. To identify these splitting points a static code analysis is performed [32]. While this enables finding splitting points for code structures fulfilling the rules defined in a knowledge base, analyzing arbitrary source code remains difficult. To tackle this issue, users are allowed to manually add annotations to their scripts, indicating suitable splitting points.

When adapting a quantum workflow based on patterns, the control flow between the different tasks is automatically redirected as discussed in Section III. However, also the data flow must be defined for the quantum workflow, i.e., it has to be specified which data is passed to and returned by the newly added tasks. The QuantME tasks define a clear interface, describing the required input and output data [9, 22]. Thus, when adding new concrete solutions to the solution repository, they must realize the interface of the corresponding QuantME tasks. For example, when adding a new concrete solution for the readout error mitigation pattern, its realization must implement the interface of the readout error mitigation task. This information can then be used during quantum algorithm augmenations (step E) to generate the data flow. In contrast, different behavioral patterns require dedicated data flow handling to rewrite the quantum workflow (step F). Therefore, each valid combination of behavioral patterns is implemented as a dedicated plugin of the QuantME framework. For example, for the prioritized execution pattern, the session identifier of the created session must be transferred to each quantum circuit execution task within the pattern sphere to use the session.

Pattern languages are typically evolving, i.e., existing patterns are adapted and new patterns are added [52]. New behavioral and enhancement patterns can be added to our approach by implementing new QuantME framework plugins or defining new injection rules, respectively. In contrast, algorithm patterns and solutions are handled generically and can directly be added to the pattern and solution repositories.

The validity of our evaluation is subject to certain threats: (i) Use case selection: While our user study showed that significant speed-ups could be achieved for the given use case, its generalizability is limited. Therefore, in future work, we plan to evaluate further use cases to investigate how the size of the workflow affects the time required for both approaches. (ii) Limited user groups: The number of participants in the study was limited. Especially the expert group had a small size, thus, the results were influenced by the participants' individual skills. (iii) Task order: As all participants solved both tasks, some speed-up due to learning effects may have occurred for the second task. However, as the participants first used our approach, the advantages could be even larger otherwise.

VII. Related Work

Various works introduce approaches for simplifying the modeling of workflows utilizing patterns. Reimann et al. [53] reduce the data complexity in simulation workflows using a set of hierarchical data management patterns that are iteratively transformed until all patterns are replaced by executable activities. Scheibler and Leymann [54] present a framework supporting the generation of workflows utilizing the enterprise integration patterns [11], which are implemented as reusable BPEL fragments. However, both approaches do not focus on the peculiarities of quantum computing and require users to define the order of all used patterns correctly. In contrast, our approach eases the modeling of the workflow by automatically placing selected patterns based on the context of the workflow.

Several works adapt workflows based on non-functional requirements or contextual changes. Bucchiarone et al. [55] and Mundbrod et al. [56] present approaches that increase the flexibility of workflows by enabling context-aware workflow adaptations, e.g., workflow fragments are selected based on the current load of available components. Further, several adaptive workflow engines, such as AgentWork [57] and AristaFlow [58], have been proposed. They enable workflows to be adapted dynamically during runtime, e.g., to modify the workflow in case of an exception or to migrate to a new version. To facilitate the utilization of hybrid runtimes for quantum workflows, Weder et al. [33] introduced an approach for rewriting hybrid loops into scripts executable by a hybrid runtime environment. Their approach is integrated as a part of step F of our method to support hybrid runtimes using patterns.

Cranganore et al. [59] present an approach to model quantum applications using scientific workflows. In the first step of their approach, the classical workflow is modeled. Subsequently, all tasks that can potentially benefit from quantum computing are complemented by an alternative quantum task. During the execution of the workflow, it is checked if the injected quantum task can be successfully executed for the given problem instance. In case it can not be executed, the classical task is executed instead. In contrast to our approach, the modeling and implementation of the quantum tasks has to be performed manually and no graphical user support is provided.

Schönberger et al. [60] investigate different quantum libraries and SDKs and evaluate the amount of quantum-specific code, as well as the complexity of porting different quantum applications to other frameworks. Thereby, they argue that the amount of quantum-specific code is limited, and thus, no further abstraction is required. However, they only focus on the code for encoding the problem and executing the quantum circuits. They do not consider techniques to improve the execution of quantum algorithms, such as warm-starting, circuit cutting, error mitigation, or the use of hybrid runtimes. However, these techniques contribute significantly to the complexity and required lines of code when realizing quantum applications. Furthermore, our user-based evaluation shows that an additional abstraction layer can provide notable speed-ups in development time while reducing complexity.

IBM uses so-called Qiskit patterns to compose functionalities developed by different stakeholders [61]. However, their definition of patterns is more abstract and rather describes the development phases of a quantum application and does not follow the pattern structure defined by Alexander et al. [10].

There are various tools for generating quantum circuits and applications. Gemeinhardt et al. [62] present a research roadmap for model-driven quantum software engineering. This research area has the goal of generating hybrid quantum applications from models. For example, Pérez-Castillo et al. [63] have shown how UML models can be used for generating quantum circuits. Alonso et al. [64] introduce a meta-model enabling the definition of quantum circuits using boolean operators which subsequently can be transformed into executable circuits. Furthermore, Gemeinhardt et al. [65] introduce a modeling language and a framework to design quantum circuits based on so-called composite operators, abstracting certain details, and to generate executable quantum circuits. Classiq presents a tool for generating quantum circuits based on a set of requirements and constraints defined in a JSON file [66]. QPath [67] is a platform for developing hybrid quantum applications aiming to support the entire quantum software lifecycle. However, the generation of the whole hybrid quantum application is not supported by any of these approaches.

Falkenthal et al. [14] present an approach to generate executable deployment models from pattern-based deployment models utilizing concrete solutions. To enable composite solutions, they introduce the concept of so-called aggregation operators, i.e., artifacts describing how two interconnected concrete solutions can be combined. Saatkamp et al. [68] present an approach to automatically determine problems within deployment models. The detected problems are resolved by adapting the deployment models utilizing suitable concrete solutions.

VIII. Conclusion & Outlook

In this paper, we presented a holistic approach for the pattern-based generation and adaptation of quantum workflows. Evaluating our approach showed a speed-up of factor 6 for non-experts and 4.5 for experts compared to the manual workflow modeling. Furthermore, by abstracting technical details and reducing complexity, our approach facilitates the development of hybrid quantum applications, particularly for non-experts. By separating the quantum-specific, reusable solutions, which are implemented by quantum experts, from the business functionality, our approach enables the development of quantum applications without deep knowledge of quantum computing.

In future work, we plan to implement the feedback obtained through the user study. Hence, we (i) plan to identify additional patterns and include them in our prototype and (ii) implement a recommendation system facilitating the selection of suitable behavioral and enhancement patterns for a given algorithm pattern, e.g., by utilizing machine learning techniques.

Acknowledgements

This work was partially funded by the BMWK projects *EniQmA* (01MQ22007B) and *SeQuenC* (01MQ22009B).

References

[1] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," *IBM Journal of Research and Development*, vol. 62, no. 6, pp. 6:1–6:20, 2018.

[2] E. P. DeBenedictis, "A Future with Quantum Machine Learning," *Computer*, vol. 51, no. 2, pp. 68–71, 2018.

[3] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

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

[5] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, Fujii *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[6] B. Weder, J. Barzen, F. Leymann, and M. Zimmermann, "Hybrid Quantum Applications Need Two Orchestrations in Superposition: A Software Architecture Perspective," in *Proceedings of the 18th IEEE International Conference on Web Services (ICWS)*. IEEE, 2021, pp. 1–13.

[7] C. A. Ellis, "Workflow technology," *Computer Supported Cooperative Work, Trends in Software Series*, vol. 7, pp. 29–54, 1999.

[8] M. Klun and P. Trkman, "Business process management–at the crossroads," *Business Process Management Journal*, vol. 24, no. 3, pp. 786–813, 2018.

[9] B. Weder, U. Breitenbücher, F. Leymann, and K. Wild, "Integrating Quantum Computing into Workflow Modeling and Execution," in *Proceedings of the 13th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2020, pp. 279–291.

[10] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[11] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2004.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley, 1994.

[13] F. Leymann, "Towards a Pattern Language for Quantum Algorithms," in *Proceedings of the First International Workshop on Quantum Technology and Optimization Problems (QTOP)*. Springer, 2019.

[14] M. Falkenthal, J. Barzen, U. Breitenbücher, and F. Leymann, "Solution Languages: Easing Pattern Composition in Different Domains," *International Journal on Advances in Software*, vol. 10, no. 3, pp. 263–274, 2017.

[15] A. Lowe, M. Medvidović, A. Hayes, L. J. O'Riordan, T. R. Bromley, J. M. Arrazola, and N. Killoran, "Fast quantum circuit cutting with randomized measurements," *Quantum*, vol. 7, p. 934, 2023.

[16] D. J. Egger, J. Mareček, and S. Woerner, "Warm-starting quantum optimization," *Quantum*, vol. 5, p. 479, 2021.

[17] OMG, *Business Process Model and Notation (BPMN) Version 2.0*, Object Management Group (OMG), 2011.

[18] OASIS, *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, Organization for the Advancement of Structured Information Standards (OASIS), 2007.

[19] J. Eder and W. Liebhart, "Workflow Transactions," *Workflow Handbook*, pp. 195–202, 1997.

[20] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, "A Survey of Data-Intensive Scientific Workflow Management," *Journal of Grid Computing*, vol. 13, no. 4, pp. 457–493, 2015.

[21] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future generation computer systems*, vol. 25, no. 5, pp. 528–540, 2009.

[22] M. Beisel, J. Barzen, M. Bechtold, F. Leymann, F. Truger, and B. Weder, "Metamodel and Formalization to Model, Transform, Deploy, and Execute Quantum Workflows," *Cloud Computing and Services Science*, vol. 1845, pp. 113–136, 2024.

[23] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, "Cost function dependent barren plateaus in shallow parametrized quantum circuits," *Nature communications*, vol. 12, no. 1, p. 1791, 2021.

[24] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, "CutQC: Using Small Quantum Computers for Large Quantum Circuit Evaluations," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '21. ACM, 2021, pp. 473–486.

[25] M. Beisel, J. Barzen, F. Leymann, F. Truger, B. Weder, and V. Yussupov, "Patterns for Quantum Error Handling," in *Proceedings of the 14th International Conference on Pervasive Patterns and Applications (PATTERNS)*. Xpert Publishing Services (XPS), 2022, pp. 22–30.

[26] J. C. Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE transactions on automatic control*, vol. 37, no. 3, pp. 332–341, 1992.

[27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2014.

[28] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," *arXiv:1411.4028*, 2014.

[29] D. Georg, J. Barzen, M. Beisel, F. Leymann, J. Obst, D. Vietz, B. Weder, and V. Yussupov, "Execution Patterns for Quantum Applications," in *Proceedings of the 18th International Conference on Software Technologies (ICSOFT)*. SciTePress, 2023, pp. 258–268.

[30] F. Truger, J. Barzen, M. Beisel, F. Leymann, and V. Yussupov, "Warm-Starting Patterns for Quantum Algorithms," in *Proceedings of the 16th International Conference on Pervasive Patterns and Applications (PATTERNS)*. Xpert Publishing Services (XPS), 2024, pp. 25–31.

[31] Kipu, "PlanQK - Pattern Atlas," 2024, https://patterns.platform.planqk.de/pattern-languages/af7780d5-1f97-4536-8da7-4194b093ab1d.

[32] D. Vietz, J. Barzen, F. Leymann, and B. Weder, "Splitting Quantum-Classical Scripts for the Generation of Quantum Workflows," in *Proceedings of the 26th Conference on Enterprise Design, Operations, and Computing (EDOC)*. Springer, 2022, pp. 255–270.

[33] B. Weder, J. Barzen, M. Beisel, and F. Leymann, "Analysis and Rewrite of Quantum Workflows: Improving the Execution of Hybrid Quantum Algorithms," in *Proceedings of the 12th International Conference on Cloud Computing and Services Science (CLOSER)*. SciTePress, 2022, pp. 38–50.

[34] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Contemporary Web Service Discovery Mechanisms," *Journal of Web Engineering*, vol. 5, no. 3, pp. 265–290, 2006.

[35] D. Weerasiri, M. C. Barukh, B. Benatallah, Q. Z. Sheng, and R. Ranjan, "A Taxonomy and Survey of Cloud Resource Orchestration Techniques," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–41, 2017.

[36] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an Encrypted and Searchable Audit Log," in *NDSS*, vol. 4. Citeseer, 2004, pp. 5–6.

[37] University of Stuttgart, "Pattern Atlas," 2024, https://github.com/PatternAtlas.

[38] ——, "QuantME Framework," 2024, https://github.com/PlanQK/workflow-modeler.

[39] ——, "Winery," 2024, https://github.com/OpenTOSCA/winery.

[40] OASIS, *Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0*, Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[41] University of Stuttgart, "OpenTOSCA Container," 2024, https://github.com/OpenTOSCA/container.

[42] ——, "Use Cases," 2024, https://github.com/UST-QuAntiL/QuantME-UseCases/tree/master/2025-icse.

[43] ——, "Demonstration Video: Pattern-based Generation and Adaptation of Quantum Workflows," https://youtu.be/6evaPun4HPA, 2024.

[44] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, p. 1115–1145, 1995.

[45] S. U. Khan, A. J. Awan, and G. Vall-Llosera, "K-Means Clustering on Noisy Intermediate Scale Quantum Computers," *arXiv:1909.12183*, 2019.

[46] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[47] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson, "The variational quantum eigensolver: A review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022.

[48] H. Mustafa, S. N. Morapakula, P. Jain, and S. Ganguly, "Variational Quantum Algorithms for Chemical Simulation and Drug Discovery," in *2022 International Conference on Trends in Quantum Computing and Emerging Business Technologies (TQCEBT)*, 2022, pp. 1–8.

[49] University of Stuttgart, "Use Case Runtime Evaluation Data," 2024, https://github.com/UST-QuAntiL/QuantME-UseCases/tree/master/2025-icse/evaluation-data.

[50] ——, "User Study Guideline," 2024, https://ust-quantil.github.io/icse-2025-evaluation.

[51] ——, "User Study Raw Data," 2024, https://github.com/UST-QuAntiL/

QuantME-UseCases/tree/master/2025-icse/user-study.

[52] R. Reiners, M. Falkenthal, D. Jugel, and A. Zimmermann, "Requirements for a Collaborative Formulation Process of Evolutionary Patterns," in *Proceedings of the 18th European conference on pattern languages of program*, 2013, pp. 1–12.

[53] P. Reimann, H. Schwarz, and B. Mitschang, "A Pattern Approach to Conquer the Data Complexity in Simulation Workflow Design," in *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*. Springer, 2014, pp. 21–38.

[54] T. Scheibler and F. Leymann, "A Framework for Executable Enterprise Application Integration Patterns," in *Enterprise Interoperability III*. Springer, 2008, pp. 485–497.

[55] A. Bucchiarone, A. Marconi, M. Pistore, and H. Raik, "Dynamic adaptation of fragment-based and context-aware business processes," in *Proceedings of the 19th International Conference on Web Services (ICWS)*, 2012, pp. 33–41.

[56] N. Mundbrod, G. Grambow, J. Kolb, and M. Reichert, "Context-Aware Process Injection: Enhancing Process Flexibility by Late Extension of Process Instances," in *On the Move to Meaningful Internet Systems: OTM 2015 Conferences*. Springer, 2015, pp. 127–145.

[57] R. Müller, U. Greiner, and E. Rahm, "AgentWork: a workflow system supporting rule-based workflow adaptation," *Data & Knowledge Engineering*, vol. 51, no. 2, pp. 223–256, 2004.

[58] S. Rinderle-Ma and M. Reichert, "Advanced Migration Strategies for Adaptive Process Management Systems," in *Proceedings of the 12th IEEE Conference on Commerce and Enterprise Computing*. IEEE, 2010, pp. 56–63.

[59] S. S. Cranganore, V. De Maio, I. Brandic, and E. Deelman, "Paving the way to hybrid quantum–classical scientific workflows," *Future Generation Computer Systems*, vol. 158, pp. 346–366, 2024.

[60] M. Schönberger, M. Franz, S. Scherzinger, and W. Mauerer, "Peel — Pile? Cross-Framework Portability of Quantum Software," in *Proceedings of the 19th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2022, pp. 164–169.

[61] IBM, "Introduction to Qiskit patterns," 2024, https://docs.quantum.ibm.com/guides/intro-to-patterns.

[62] F. Gemeinhardt, A. Garmendia, and M. Wimmer, "Towards Model-Driven Quantum Software Engineering," in *Proceedings of the 2nd IEEE/ACM International Workshop on Quantum Software Engineering (Q-SE)*. IEEE, 2021, pp. 13–15.

[63] R. Pérez-Castillo, L. Jiménez-Navajas, I. Cantalejo, and M. Piattini, "Generation of Classical-Quantum Code from UML models," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 2. IEEE, 2023, pp. 165–168.

[64] D. Alonso, P. Sánchez, and F. Sánchez-Rubio, "Engineering the development of quantum programs: Application to the Boolean satisfiability problem," *Advances in Engineering Software*, vol. 173, p. 103216, 2022.

[65] F. Gemeinhardt, A. Garmendia, M. Wimmer, and R. Wille, "A Model-Driven Framework for Composition-Based Quantum Circuit Design," *ACM Transactions on Quantum Computing*, vol. 5, no. 4, pp. 1–36, 2024.

[66] N. Minerbi, "Quantum software development with classiq," in *Quantum Software Engineering*. Springer, 2022, pp. 269–280.

[67] Q. S. Technologies, "Quantumpath," 2024, https://www.quantumpath.es/.

[68] K. Saatkamp, U. Breitenbücher, M. Falkenthal, L. Harzenetter, and F. Leymann, "An Approach to Determine & Apply Solutions to Solve Detected Problems in Restructured Deployment Models Using First-Order Logic," in *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER)*. SciTePress, 2019, pp. 495–506.

All links were last followed on February 3, 2025.