Institute of Architecture of Application Systems

MODULO: Modeling, Transformation, and Deployment of Quantum Workflows

Benjamin Weder, Johanna Barzen, and Frank Leymann

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{weder, barzen, leymann}@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{Weder2021_MODULO,
  author    = {Weder, Benjamin and Barzen, Johanna and Leymann, Frank},
  title     = {{MODULO: Modeling, Transformation, and Deployment
               of Quantum Workflows}},
  booktitle = {Proceedings of the 25th IEEE International
               Enterprise Distributed Object Computing Workshop (EDOCW 2021)},
  year      = {2021},
  month     = oct,
  pages     = {341--344},
  doi       = {10.1109/EDOCW52865.2021.00067},
  publisher = {IEEE Computer Society}
}
```


Universität Stuttgart
Germany

MODULO: Modeling, Transformation, and Deployment of Quantum Workflows

Benjamin Weder, Johanna Barzen, and Frank Leymann
University of Stuttgart, Institute of Architecture of Application Systems, Germany
Email: [firstname.lastname]@iaas.uni-stuttgart.de

***Abstract*—Quantum applications are usually hybrid, i.e., they require executing quantum programs and classical programs, e.g., performing pre- or post-processing tasks. To benefit from advantages, such as robustness, reliability, or scalability, these programs can be orchestrated using quantum workflows. However, modeling quantum-specific tasks in workflows is complex and requires deep technical and mathematical knowledge. Furthermore, additional steps have to be performed before executing the workflow, e.g., the deployment and binding of the needed services. In this demonstration, we present the MODULO framework, providing a workflow modeling extension easing the modeling of quantum workflows. It comprises an integrated toolchain to graphically model quantum workflows, transform and package them in a self-contained archive, and automatically deploying the workflows together with their required services.**

***Index Terms*—Quantum Computing, Workflow Technology, Quantum Workflows, Service Deployment Automation**

I. Introduction and Motivation

Recent advances in developing new or improving existing quantum computers enable utilizing quantum computing in more and more application areas, e.g., machine learning or simulations [1], [2]. Furthermore, quantum computers from different vendors, such as IBM or Rigetti, are publicly accessible via the cloud [3], [4]. Thus, the need for developing quantum applications will increase dramatically in the next years. Such quantum applications are typically hybrid, consisting of quantum and classical programs [5], [6]. Classical programs are, e.g., used to prepare classical data for the processing in the quantum computer or mitigate errors after performing a quantum computation, which is needed due to today's noisy quantum computers [6], [7]. Additionally, different quantum algorithms, such as *Simon's* or *Shor's algorithm*, require algorithm-specific pre- and post-processing steps, which have to be implemented using classical programs [2], [5]. Therefore, the programs comprising a quantum application must be orchestrated, which can be done using so-called *quantum workflows* [5]. The usage of mature workflow systems enables benefiting from their robustness, scalability, and sophisticated error handling mechanisms [8]. Further, quantum workflows ease the integration of quantum applications into more complex business applications relying on workflow technology.

However, existing workflow languages do not provide explicit modeling constructs for the required pre-processing, quantum program execution, and post-processing tasks [7]. This leads to a complex and error-prone modeling process of quantum workflows, requiring deep technical and mathematical knowledge. The missing explicit modeling constructs also complicate finding existing implementations for different tasks to reuse them. Additionally, before executing a quantum workflow, the required services invoked by the workflow must be deployed and bound to the workflow [9]. However, quantum experts implementing a quantum application are often not used to these tasks. Thus, they must be automated as far as possible.

To tackle these challenges, we present the *MODULO framework*, enabling the *modeling, transformation, and deployment of quantum workflows*. It incorporates a workflow modeling extension for quantum computing with a graphical notation to ease the modeling of quantum workflows. Further, it provides an integrated toolchain supporting the automated transformation of resulting workflow models to native workflow models to retain their portability, as well as the deployment of the quantum workflow with the required services and their binding. In the following, we introduce the method implemented by the MODULO framework and the corresponding system architecture. Finally, we summarize our demonstration, including the target audience, useful takeaways, and two use cases.

II. Modeling, Transformation, and Deployment of Quantum Workflows

In this section, we present our method to model, transform, and deploy quantum workflows, which is supported by the MODULO framework. Fig. 1 gives an overview of the method.

A. QuantME Modeling

First, the quantum workflow is modeled by defining the collection of required activities, as well as their partial order and data flow between them in a workflow model [5], [8]. Thereby, the activities, e.g., comprise different pre- and post-processing tasks, the execution of quantum programs, or other traditional activities, such as human tasks or service invocations [5]. To ease the modeling of quantum-specific tasks and increase the reuse of their implementations, we introduced the *Quantum Modeling Extension (QuantME)* [7], providing explicit modeling constructs for different frequently occurring tasks. Furthermore, each modeling construct specifies a set of configuration attributes to customize it depending on the context in the quantum workflow, e.g., by defining the quantum computer for the execution of a quantum program [7]. Thereby, QuantME can be applied to various imperative workflow languages, such as the *Business Process Model and Notation (BPMN)* [10], which is also used within the MODULO framework.

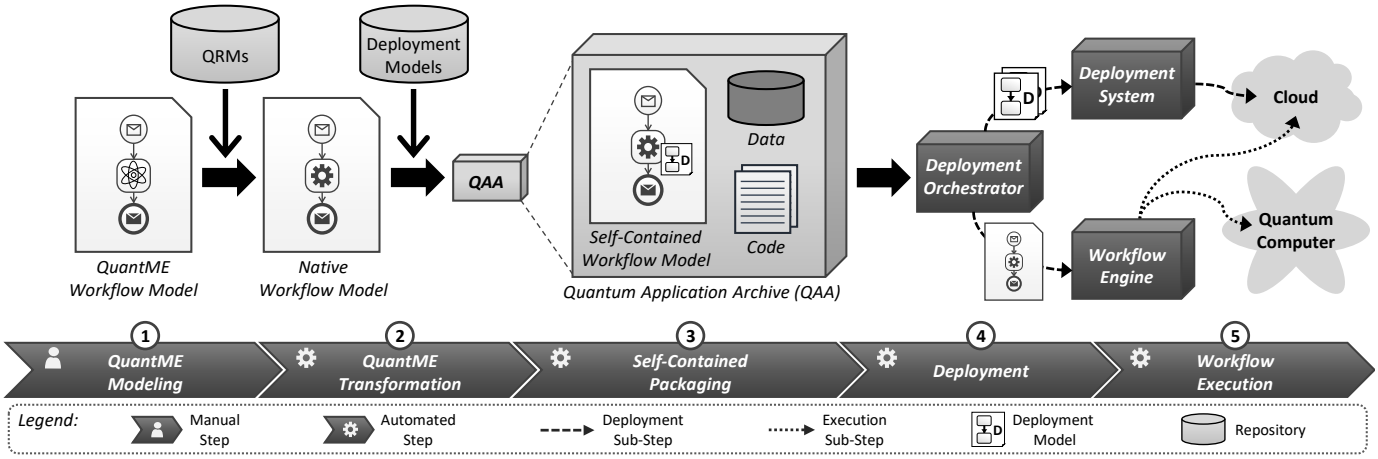

Fig. 1. Overview of the MODULO method, supporting the modeling, transformation, and deployment of quantum workflows

B. QuantME Transformation

However, the usage of the QuantME modeling constructs reduces the portability of the workflow models between different workflow engines, as the target workflow engine must be extended to support the new modeling constructs [7]. Therefore, the workflow model is transformed into a native workflow model in the second step, consisting only of native modeling constructs of the extended workflow language. For this, the QuantME modeling constructs are iteratively replaced by reusable *workflow fragments* implementing the required functionality [7], [11]. Thereby, so-called *QuantME Replacement Models (QRMs)* [7] are utilized (see Fig. 1). In addition to the workflow fragments, QRMs contain a *detector*, defining the QuantME modeling construct that can be replaced. Furthermore, restrictions can be specified, e.g., that the QuantME modeling construct can only be replaced if it is configured in a certain way utilizing its configuration attributes. For example, if a QRM is only suitable to execute quantum programs using quantum computers from IBM and not from Rigetti, this can then be defined in the attributes of the detector. Thereby, the workflow fragments used for the replacement are also allowed to contain QuantME modeling constructs. Hence, these modeling constructs are replaced in the next iterations until a native workflow model is reached.

C. Self-Contained Packaging

After the transformation, the native workflow model can be transferred into the target environment for its execution. However, the manual transfer of all needed information, e.g., the required software artifacts to deploy a service that is invoked by the workflow, is time-consuming and error-prone [9]. Therefore, this information is packaged into a self-contained archive, the so-called *quantum application archive (QAA)* [5]. Hence, only a single archive has to be transferred into the target environment. The QAA comprises the *self-contained workflow model* [9], which has a deployment model attached to each activity of the workflow model that requires the deployment of a service. These services can then be automatically deployed in the target environment before executing the workflow (see Section II-D). Furthermore, it also allows attaching multiple deployment models to a workflow activity. This enables the selection of one deployment model, e.g., based on the available infrastructure in the target environment or the quality of service requirements. Additionally, the QAA contains the code of required quantum and classical programs, as well as special data if needed by the quantum workflow [5].

D. Deployment

In the target environment, the QAA is passed to the so-called *deployment orchestrator* [9]. The deployment orchestrator is in charge of deploying required services for quantum workflows. Thus, it extracts the deployment models from the QAA and passes them to a corresponding deployment system. Thereby, the deployment orchestrator is plugin-based to support using different *deployment systems*, such as Kubernetes [12], Terraform [13], or the OpenTOSCA Container [14], depending on the type of deployment models. If multiple deployment models are defined for a certain activity, the deployment orchestrator automatically selects a suitable one [9]. Most quantum computers offered over the cloud require sending quantum programs together with the execution request [4]. Then, no previous deployment of these programs is needed. After deploying all required services, the binding between workflow and services must be performed. For this, the deployment orchestrator enriches the workflow model with information about the service endpoints, such as the protocol to use to invoke the service or its IP address. Afterwards, the workflow is uploaded to a *workflow engine* for execution.

E. Workflow Execution

Finally, the quantum workflow is instantiated and executed. The instantiation can be triggered by a certain event, e.g., periodically executing the quantum workflow, or by the user, who can then pass additional input parameters to the workflow instance [7], [9]. During workflow execution, the workflow engine invokes deployed services and kicks off the execution of quantum programs on a quantum computer (see Fig. 1) [5].

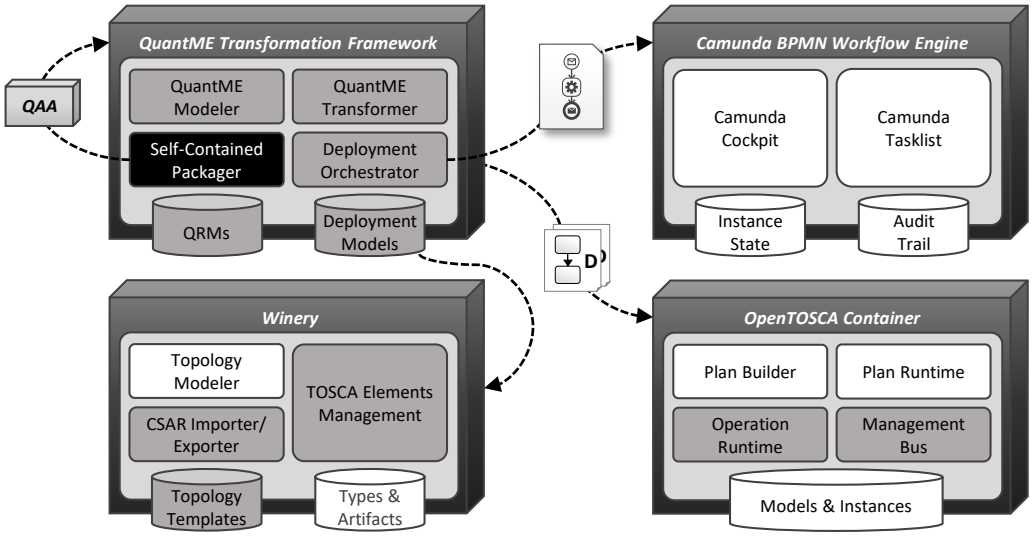

Fig. 2. Architecture of the MODULO framework (new components are black, extended components are grey, and existing unchanged components are light)

III. Overview of the MODULO Framework

The MODULO framework is an extension of the *Camunda workflow system* [1], consisting of a graphical BPMN modeler and a BPMN workflow engine, as well as extensions of the *OpenTOSCA ecosystem* [14]. The OpenTOSCA ecosystem comprises *Winery*, a graphical modeling tool for TOSCA-based deployment models, and the *OpenTOSCA Container*, a TOSCA-compliant deployment system. Fig. 2 shows the architecture of the framework with the relevant components.

The *QuantME Transformation Framework*[2] extends the Camunda BPMN modeler to support the QuantME modeling constructs. Therefore, the *QuantME modeler* component enables the graphical modeling of all introduced modeling constructs, as well as the native BPMN modeling constructs [7]. Furthermore, the transformation to native workflow models is implemented by the *QuantME transformer*, using a corresponding database with *QRMs* (see Section II-B). The *self-contained packager* component is in charge of packaging all required information into the self-contained QAA (see Section II-C) [5]. For this, it uses a repository with *deployment models* that can be associated with activities in the workflow model, which are then added to the archive on export [9]. This repository is implemented by a Winery instance, managing the available TOSCA-based deployment models [14], [15]. Finally, the *deployment orchestrator* performs the deployment of all required software artifacts for a quantum workflow and the binding of the workflow and the deployed services [9].

The *Camunda BPMN Workflow Engine* is used without extensions in the MODULO framework, as the workflow models are transformed to native workflow models before uploading and executing them [7]. It comprises the *Camunda cockpit*, visualizing the uploaded workflow models, the corresponding workflow instances, and their current state. Further, pending user tasks of the workflow instances are managed by the *Camunda tasklist*. While a workflow instance is running, its state is stored in the *instance state* database, e.g., the currently active activities [5]. After termination of the instance, this information is moved to the *audit trail* for long-term storage.

Winery[3] provides a *topology modeler* component, enabling the graphical modeling of topologies for applications. Topologies describe the components comprising an application and their desired target state [9]. Hence, they are also referred to as declarative deployment models and can be used to deploy the required services for a quantum workflow [5]. As Winery is based on the TOSCA standard [15], topologies are defined as *topology templates* and stored in a corresponding repository [14]. However, topology templates usually depend on different TOSCA elements, e.g., node types, to define the semantics of a component [15]. Therefore, these elements are managed by the *TOSCA elements management* component and stored in the corresponding *types & artifacts* repository within Winery. Finally, before attaching the required topology templates to the activities of a quantum workflow, they have to be packaged with all their dependencies. For this, the TOSCA standard introduces the *cloud service archive (CSAR)* packaging format. Thus, the *CSAR importer/exporter* component enables packaging, exporting, and importing CSARs.

All CSARs attached to activities of the workflow model in the QAA are uploaded to the *OpenTOSCA Container*[4] by the deployment orchestrator. Then, the *plan builder* interprets the contained declarative topology template and derives an imperative provisioning plan defining the deployment operations to be executed to set up an application. For the deployment, the provisioning plan is executed using the *plan runtime* component. During runtime, the plan invokes different software artifacts implementing the required deployment operations,

[1]https://camunda.com/products/camunda-platform
[2]https://github.com/UST-QuAntiL/QuantME-TransformationFramework
[3]https://github.com/OpenTOSCA/winery
[4]https://github.com/OpenTOSCA/container

e.g., to create a virtual machine. These software artifacts can be implemented, e.g., as a WAR file using Java or by a Python script. Hence, a corresponding runtime for the various kinds of software artifacts is required, which is provided by the plugin-based *operation runtime*. Further, different software artifacts might require varying protocols or invocation mechanisms. These details are abstracted by the *management bus*, which handles all communication. Finally, the uploaded CSARs and information about service instances that are currently provisioned or for which the deployment is already finished are stored in the *models & instances* repository.

IV. Demonstration

In the following, we discuss the target audience of this demonstration and how they can benefit from MODULO. Additionally, we present the use cases of the demonstration.

A. Target Audience & Benefits of the MODULO Framework

The MODULO framework is intended for two different user groups, and thus, also the target audience that will benefit from attending the demonstration is twofold: (i) First, workflow modelers that are used to workflow technologies and want to integrate quantum computations into their workflows, but have only limited knowledge about quantum computing. Due to the abstractions provided by QuantME, they can model the different pre-processing, quantum program execution, and post-processing tasks of a quantum algorithm without having to understand all the quantum-specific details. Furthermore, the explicit configuration attributes for the various QuantME tasks still enable them to customize the steps of the quantum algorithm to their needs, e.g., by selecting different data encodings or quantum computers to use. (ii) The second user group are quantum experts who often implement their quantum applications as monolithic python applications and want to incorporate the advantages provided by mature workflow technologies into their applications. Thereby, they also benefit from the introduced modeling extension, but additionally, the automation of the service deployment and binding is important for them to ease the usage of workflows. Finally, both user groups will take advantage of the reduced modeling effort and the increased reuse of programs, services, and workflow fragments, which we evaluated in previous work [7].

B. Use Cases & Presented Features

To demonstrate the MODULO method presented in Fig. 1, we utilize two different use cases that are detailed below.

Simon's algorithm: First, the various QuantME modeling constructs [7], their purpose, and their graphical notation are introduced. These modeling constructs are then used to model a workflow implementing Simon's algorithm using the MODULO framework. Thereby, it can be modeled abstractly using just one QuantME task and is then transformed in two steps into a native workflow model comprising seven tasks, showing the achieved simplification. Further, it is demonstrated how different configuration attributes of the QuantME tasks result in different workflow models during transformation.

Quantum machine learning: Although the first use case implementing Simon's algorithm provides a comparatively simple workflow model, which is easy to understand, the practical applicability of the solved problem is limited. Thus, the second use case demonstrates a more complex quantum workflow executing two quantum algorithms for clustering and classification [5]. For this use case, it is shown how deployment models for services are modeled, attached to activities of a workflow, and how the MODULO framework supports their automated deployment and binding with the workflow.

The discussed use-cases, as well as further example use cases implemented using the MODULO framework, can be found on Github [16], together with a detailed description of how to set up and use the framework. Furthermore, a short demonstration video is available on YouTube [17].

Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in *Simulation Technology* (EXC 2075 – 390740016) at the University of Stuttgart. This work was partially funded by the BMWi project *PlanQK* (01MK20005N).

References

[1] J. Barzen, "From Digital Humanities to Quantum Humanities: Potentials and Applications," *arXiv:2103.11825*, 2021.
[2] B. Weder *et al.*, "Quantum Software Development Lifecycle," *arXiv:2106.09323*, 2021.
[3] F. Leymann *et al.*, "Quantum in the Cloud: Application Potentials and Research Opportunities," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER)*. SciTePress, 2020, pp. 9–24.
[4] R. LaRose, "Overview and Comparison of Gate Level Quantum Software Platforms," *Quantum*, vol. 3, 2019.
[5] B. Weder *et al.*, "Hybrid Quantum Applications Need Two Orchestrations in Superposition: A Software Architecture Perspective," in *Proceedings of the 18th IEEE International Conference on Web Services*. IEEE, 2021.
[6] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the NISQ era," *Quantum Science and Technology*, vol. 5, no. 4, 2020.
[7] B. Weder *et al.*, "Integrating Quantum Computing into Workflow Modeling and Execution," in *Proceedings of the 13th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2020, pp. 279–291.
[8] F. Leymann and D. Roller, *Production Workflow: Concepts and Techniques*. Prentice Hall PTR, 2000.
[9] B. Weder *et al.*, "Deployable Self-contained Workflow Models," in *Proceedings of the 8th European Conference on Service-Oriented and Cloud Computing (ESOCC)*. Springer, 2020, pp. 85–96.
[10] OMG, *Business Process Model and Notation (BPMN) Version 2.0*, Object Management Group, 2011.
[11] H. Eberle *et al.*, "Process Fragments," in *On the Move to Meaningful Internet Systems (OTM)*. Springer, 2009, pp. 398–405.
[12] CNCF. (2021) Kubernetes. [Online]. Available: https://kubernetes.io
[13] HashiCorp. (2021) Terraform. [Online]. Available: https://www.terraform.io
[14] U. Breitenbücher *et al.*, "The OpenTOSCA Ecosystem - Concepts & Tools," *European space project on smart systems, big data, future internet-Towards serving the grand societal challenges*, vol. 1, pp. 112–130, 2016.
[15] OASIS, *Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0*, 2013.
[16] University of Stuttgart. (2021) Quantum Workflow Use Cases. [Online]. Available: https://github.com/UST-QuAntiL/QuantME-UseCases
[17] University of Stuttgart. (2021) Demonstration Video. [Online]. Available: https://youtu.be/7MveQFcfkDQ

All links were last followed on September 24, 2021.