Computing (2022) 104:2375–2403
https://doi.org/10.1007/s00607-022-01091-4

REGULAR PAPER

Design of classical-quantum systems with UML

Ricardo Pérez-Castillo[1,2] · **Mario Piattini**[2]

Received: 28 September 2021 / Accepted: 5 May 2022 / Published online: 31 May 2022

Abstract

Developers of the many promising quantum computing applications that currently exist are urging companies in many different sectors seriously consider integrating this new technology into their business. For these applications to function, not only are quantum computers required, but quantum software also. Accordingly, quantum software engineering has become an important research field, in that it attempts to apply or adapt existing methods and techniques (or propose new ones) for the analysis, design, coding, and testing of quantum software, as well as playing a key role in ensuring quality in large-scale productions. The design of quantum software nevertheless poses two main challenges: the modelling of software quantum elements must be done in high-level modelling languages; and the need to further develop so-called "hybrid information systems", which combine quantum and classical software. To address these challenges, we first propose a quantum UML profile for analysing and designing hybrid information systems; we then demonstrate its applicability through various structural and behavioural diagrams such as use case, class, sequence, activity, and deployment. In comparison to certain other quantum domain-specific languages, this UML profile ensures compliance with a well-known international standard that is supported by many tools and is followed by an extensive community.

Keywords Quantum computing · Quantum software engineering · Software design · Quantum modelling · UML

Mathematics Subject Classification 81-04 · 68N19

✉ Ricardo Pérez-Castillo
ricardo.pdelcastillo@uclm.es

Mario Piattini
mario.piattini@uclm.es

[1] Faculty of Social Sciences and IT, UCLM, Av. Real Fábrica de Seda s/n, 45600 Talavera de la Reina, Spain

[2] Information Technology and Systems Institute, UCLM, Camino Moledores s/n, 13005 Ciudad Real, Spain

1 Introduction

Quantum computing is destined to revolutionise how society will solve computationally challenging problems, i.e., some problems that cannot be addressed by classical computers in a practical, reasonable time frame. Quantum computing is, therefore, gaining increasing relevance due to its promising applications in many business domains, e.g., financial services [1] and the economy [2], chemistry simulation and molecular design [3], medicine and drug development [4], information security [5], supply chains and logistics [6], artificial intelligence [7], among many others.

Notwithstanding preliminary demonstrations of such advances and their potential, the advantages offered by quantum computing cannot be realised through the use of cutting-edge quantum computers in isolation, but rather quantum software is also required, and this will undoubtedly play an important role [8, 9]. Definitely, "software is the invisible writing that whispers the stories of possibility into our hardware" [10].

The quantum software programming techniques we have as of today have been experimentally proposed in an ad hoc manner. Consequently, Quantum Software Engineering (QSE) [11] has emerged as a new research field aimed at producing quantum software by applying the knowledge and lessons learned from the classical software engineering field. This requires the application or adaptation of existing software engineering processes, methods, techniques, practices, and principles – or the creation of new ones – to the development of quantum software. In the field of QSE, one of the problems still to be resolved is how to overcome the lack of techniques for the analysis and design of whole classical-quantum information systems. The contribution of this paper is to attempt to fill this gap.

Up until now, most quantum software has been coded with no previous analysis of the problem or the design models. Requirement engineering and conceptual modelling, however, have demonstrated that they can help in this matter [11]. Nevertheless, there are two challenges that currently prevent a simple application to quantum software of the existing analysis and design techniques

- *Quantum software modelling* Quantum computing is based on the counter-intuitive principles of quantum mechanics [12], including superposition and entanglement, among others. Thus, the lessons learned from existing and classic design techniques cannot simply be applied as they are [13] but rather new quantum foundations and features need to be conceived. In this regard, quantum software is still currently being coded in a manual, *ad hoc* way. For example, although there are some well-known algorithms that are defined in the literature through mathematic formalisms, those algorithms have been manually coded in multiple ways for specific quantum programming languages without any prior design or modelling.
- *Hybrid quantum information systems* The further adoption of quantum information systems will not entail the complete replacement of classic information systems. Both technologies will operate in parallel in what are known as "hybrid information systems" [14]. This is because it does not make sense to use quantum software to solve simple and simplistic problems, since these are already adequately performed by classical computers, and at less cost. Thus, classical software will in future act as a driver that will orchestrate requests to quantum software. Classical-quantum

information systems must therefore be analysed and designed together, which in itself poses another important challenge.

Notwithstanding these two challenges, many of the problems currently solved by existing classical software engineering methods and techniques are still the same as those we find in the design and construction of quantum software [15]. For example, abstract descriptions for software are key for (i) discussing design concerns, and (ii) modelling systems with high-level, conceptual representations while the implementation details are hidden. One tried and tested solution, which is the most widely used in classical software engineering, is the use of standard modelling languages—such as the Unified Modelling Language (UML) [16]—for the analysis and design of information systems. The chief hypothesis behind this proposal is that UML modelling should be seen as useful for the analysis and design of hybrid information systems.

The main contribution of this paper is to provide a quantum UML profile that covers the main modelling aspects of the analysis and design of hybrid information systems. The quantum UML profile also covers structural and behavioural aspects. In particular, it addresses use cases, class, sequence, activity, and deployment diagrams. As a result, both quantum and classical elements, as well as the relationships between them, can be modelled together in an integrated design.

Although in previous proposals UML has been extended through various Domain-Specific Languages (DSL) [17], this proposal extends it by means of a profile that is the standard mechanism for this purpose. Although in comparison with DSLs the expressiveness of a quantum extension implemented in a UML profile is limited, the main implication of this extensibility strategy is that the outgoing design models are still UML-compliant [18]. Consequently, it is not necessary to adapt either UML tools or the training given to designers, since the notation is well-known and supported by many tools.

The rest of this paper is structured as follows. Section 2 summarises the current state of the art, while Sect. 3 discusses the relevant academic work. Section 4 presents the main proposal, the quantum UML profile with some examples of UML diagrams. Section 5 discusses applicability and implications for researchers and practitioners. Section 6 provides an example of application, where a hybrid application is built using the quantum UML Profile. Finally, Sect. 7 provides conclusions, along with the future work.

2 The state of the art

This section provides the underlying principles necessary to understand the main proposal. First, quantum computing is introduced in Sect. 2.1; then hybrid classical-quantum information systems and the challenges they pose are explained in Sect. 2.2; finally, the emerging field of QSE is presented in Sect. 2.3 together with the issues that remain to be resolved.

2.1 Principles of quantum computing

Classical computers store and handle information in bits (i.e., 0 and 1), while quantum computers introduce the notion of the qubit (quantum bit), which is modelled by a probabilistic function in which the states 0 and 1 have, at the same time, a certain probability [19]. Underlying this is the superposition principle of quantum mechanics. Depending on the quantum computing technology involved, qubits are internally managed by a polarised photon or by an electron spin, among others. The actual state of a qubit is only known once it is measured, although in that moment the qubit collapses and therefore its state cannot be changed anymore. Superposition and the probabilistic nature of qubits (as well as other counterintuitive principles like entanglement or no cloning) make quantum algorithms very different to the algorithms used in classical computing. While classical algorithms are based on a sequence of imperative steps to manipulate data in a deterministic way, quantum algorithms function differently. Quantum software deals with uncertainty and attempts to achieve solutions as optimal as possible by exploring the search space based on probabilistic functions [12]. The mechanisms to alter the probabilities of states 0 and 1 are known as the "quantum gates", whose application under various qubits is combined in quantum circuits that represent quantum algorithms (or parts there that can be encapsulated in new gates).

Quantum computing technology and quantum programming languages present various differences compared to classical versions [20]: (i) a lack of control structures that can manage traditional control in addition to the non-deterministic and probabilistic ones; (ii) a deficient derivation of quantum programs in a compositional manner; and (iii) the absence of formulation of interfaces and signatures based on parameters, that are then computed dynamically. These differences allied to the intrinsic nature of quantum software, hinder the application of the well-known, existing programming techniques.

2.2 Hybrid quantum information systems

Quantum computing is not an all-in-one technology. Due to the intrinsic nature of quantum computing, it can be suitable to address a particular kind of problem, but not every single one. For some problems, quantum computers suffer from the same limitations as classical ones [21], while at other times if the task is quite simple there is no reason to use quantum computing (a more expensive technology) to address what classical computers can still perform quite successfully. As a result, in the future, modern information systems will probably integrate classic and quantum software [14] in so-called "hybrid information systems". These systems support some business operations implemented with classical programming languages which perform calls to quantum algorithms that can address certain otherwise intractable problems. In these hybrid systems classical software runs in classical computers (the controller or driver part) while quantum software runs in quantum computers (the responder part) that are typically located in the cloud. Since today's computational power for NISQ (Noisy Intermediate-Scale Quantum) devices is still limited to a degree, several NISQ devices could be used in a distributed quantum computing architecture [22]. Hybrid

information systems use the classical controller part for supporting two important tasks: the cost function and the optimisation [23]. The cost function refers to the part in which the quantum answer (based on the qubits' measures) is translated into a meaningful answer to a real-world problem. The optimisation, among other tasks, is concerned with the estimation and control of how many executions of the quantum part are needed to get a reliable answer, since it is based on a probabilistic search space. According to [24], hybrid information systems face obstacles in code portability, tool integration, program validation, and in the orchestration of workflow development. For one of these obstacles, code portability, there are already some representations like QIR (Quantum Intermediate Representation) [25].

2.3 Quantum software engineering

The radical differences between quantum and classical software, alongside the fact that "quantum computer tools are in their infancy" [26], have led mathematicians, physicists, and computer scientists to code quantum software by following distinct, ad hoc methods [27, 28]. Until a few years ago, this non-systematic way for coding quantum software was sufficient for experimental quantum software [11]. However, with the advent of "industrial" quantum software and its increasing use in various business domains, there is a growing demand for quantum software to be produced in a more systematic way, all the while meeting quality and other budget and time constraints [9]. Thus, quantum software is nowadays not only coded but is also being analysed, designed, verified, etc. throughout a whole lifecycle [15, 29]. A survey of the current research initiatives into the different phases of the lifecycle of quantum software can be found in [30]. Also, it should be noted that specific quantum software tooling is necessary to cope with these demands [31]. It is in this context that the QSE field has developed, taking lessons learned from classic computing in the way of techniques, methods, and practices [20]. Such techniques and methods have been demonstrated to deal with well-known challenges, also shared by quantum software, through the systematic application of the following principles [10]: craft sound abstractions, maintain a clear separation of concerns, strive for a balanced distribution of responsibilities, and always seek simplicity. The community now has the chance to get theory and techniques in place before quantum software emerges further and becomes heavily popularised.

Among other challenges in the QSE field are the need for conceptual designs and software patterns, metrics and quality assurance, reliable testing and verification, as well as software maintenance and evolution [11]. This paper focuses on one of these challenges: the analysis and design of quantum software and, in general, of hybrid information systems.

3 Related work on quantum software design

Within QSE, the analysis and design of quantum software deserve special attention. Software analysis deals with the problem domain, i.e., what the problem is. Design,

meanwhile, is associated with the solution domain, i.e., how the problem should be solved. Below we set out some of the initiatives that deal with quantum design.

Most of the quantum design efforts to date have been made in the context of quantum algorithms. For example, design patterns to translate boolean, reversible logic into quantum circuits based on quantum gates [32]. Similarly, Wang et al. [33] propose the qFBE method for the reversible implementation of algebraic functions. Wille et al. [34] also studied the Computer-Aided Design (CAD) methods and tools for quantum data structures and algorithm mappings for different quantum architectures. Genç et al. [35] provide a quantum composer tool for designing quantum algorithms through a virtual reality environment. All these methods help to code quantum software, although as design solutions they do not contribute to the analysis and design of whole hybrid information systems.

Other design techniques, known as mapping techniques, have been developed to adapt hardware-agnostic quantum circuits to the specific restrictions of quantum computers. For example, Bandic et al. [36] propose a mapping technique based on a structured design for space exploration strategy. Zhou et al. [37], meanwhile, suggest a mapping technique based on simulated annealing and heuristic search. In a similar way to the previously mentioned group of design methods, mapping techniques cannot be considered as high-level design solutions, i.e., those managing abstract, conceptual elements.

There are other methods and techniques that are not really design solutions, but which nevertheless provide some abstraction layers to hide low-level coding details. For example, Thompson et al. [38] provide a non-algorithmic machine learning approach that avoids the need to break down an algorithm into quantum gates, thus reducing unnecessary complexity. However, this method is limited by the dependency on training sets that by definition can bias the outcomes of the method. Another low-code design method, known as *LIQUi*, is presented in [39], which provides a DSL for optimisation, rendering, or translation of quantum circuits. Chancellor et al. [40] provide a decision-making framework for use cases of quantum and quantum-inspired algorithms. This work focuses on the methodological part to decide if a use case should, or should not, be implemented as quantum software. It does not, however, provide support for modelling quantum use cases.

For the design of hybrid information systems, Weder et al. [41] propose *QuantMe* as a generic extension for imperative workflow modelling languages. This proposal enables the integration of quantum computations and can model the orchestration of classical applications and quantum circuits. The author applies the extension with BPMN (Business Process Model and Notation) in *Quantum4BPMN* [42], an application that allows one to use BPMN notation in the workflow design of hybrid information systems. This work focuses on the execution workflow of quantum software within hybrid information systems, however the abstract, conceptual design is outside of the scope of that proposal. Exman and Shmilovich [43] also propose a technique that works for both classical and quantum software, which uses a density matrix for defining formal modular designs.

Other works deal with conceptual designs, both for the static (structural) and the dynamic (behavioural) parts. For example, Ali and Yue [44] present guidelines for developing quantum software modelling languages and provide a conceptual model of

quantum programs as well as an example of modelling based on state machines. Perez-Delgado and Perez-Gonzalez [17] propose a Q-UML extension for modelling quantum elements together with classical elements. This extension provides specific graphical notations for class and sequence diagrams, such that it is defined as a fully independent extension. Along similar lines, Pérez-Castillo et al. [45] propose a preliminary UML extension for defining quantum circuits based on UML-compliant activity diagrams. That work focuses on quantum circuits modelling that are defined by using a UML profile and is, therefore, compliant with existing UML tools. Comparing these two last-mentioned proposals, the previous proposal in [45] does not cover other UML diagrams and conceptual design apart from quantum circuits as the current work attempts.

4 Quantum UML profile

This section presents the main contribution of the paper. First, Sect. 4.1 justifies the selection of the quantum UML profiles against other alternatives. Then, Sects. 4.2–4.6 explain in detail the quantum UML profile which we have used for facilitating the analysis and design of hybrid information systems.

4.1 Rationale of the proposal

The development of hybrid information systems should follow a complete life cycle, i.e., analysis, design, coding, testing, and so forth [15, 29]. Irrespective of the precise nature of the life cycle is, the quantum software has to be designed at some point. As shown in the section dealing with related work, the development of quantum software, whether isolated or within hybrid information systems, has usually been accomplished through the direct implementation of code modules. Only in some preliminary proposals, some abstract specifications have been used prior to the coding stage [17, 45]. However, the modelling methods that have been proposed up to now are not fully UML-compliant (and therefore are not ready to be used with the existing tools), and none of the modelling methods yet cover all of the design concerns.

In general terms, software design serves to integrate definitions for the architecture, system elements, interfaces and other characteristics of a system [46], seeking to accomplish goals using a set of primitive components and meeting certain constraints [47]. UML has thus far proven to be a useful means to define classical software designs [48]. UML can help us by gathering and analysing software requirements and incorporating them into a program design in a technology- and methodology-independent manner [49].

Due to the well-proven benefits of using UML in classical software design, we suggest it is also possible to use UML in designing hybrid information systems. Although other modelling languages may be used to design software, we believe that the use of UML in quantum software engineering delivers several benefits:

- *Different concern viewpoints* UML provides different kinds of diagrams with which to look at information systems from varying perspectives and to repre-

sent the views of different concerns. It is useful to take into account these different viewpoints for modelling hybrid information systems, e.g., to distinguish quantum software pieces from classical counterparts.

- *Design validation* These various perspectives help quantum software engineers to explore, communicate and validate alternative designs. The use of UML is highly extended in industry these days and is reasonably easy to understand for non-technical staff. For hybrid information systems, this is useful since professionals apart from software engineers can understand and validate designs.
- *Best practices* The use of UML inherently represents a collection of the best engineering practices that have proved successful in the modelling of large complex information systems. These practices could consequently be applied in quantum/hybrid information systems as well. One example of this is the application of the Model-Driven Engineering (MDE) approach [50] to ensure platform independence and the functioning of low-code, generative techniques.
- *Organised design* UML modelling makes it simpler to organise software as a collection of self-contained modules and components. The structured decomposition improves the reuse of code, scalability, maintainability, robustness, among others. In fact, these features are now demanded by the quantum software engineering community [9].
- *Tooling* Since UML is an ISO/IEC standard (firstly released by OMG) and is widely adopted, most of the existing design and modelling tools already support it. In fact, one of the goals for UML is to advance the state of industry by enabling object visual modelling tool interoperability [16]. As a result of this, quantum software modelling could be integrated into the existing tools. For example, other DSL and specific modelling techniques preliminary used for quantum software do not ensure that integration with the existing tools. As a result, this UML feature ensures its applicability.
- *Learning curve* UML is already used by many software engineers, so its extension into quantum fields could be learned without any great need for additional training.
- *Software modernisation* In addition to the design of target hybrid systems that are then implemented by forward engineering, UML models are also used as a notation for the abstract representations obtained from the source code by reverse engineering tools. Thus, UML serves as a means to migrate or modernise classical and quantum software towards hybrid information systems [14].

Despite such benefits, however, UML still needs to be adapted in order to capture all the new semantics and building elements involved in the development of quantum software, as has been shown in some of the previous work undertaken (cf. Sect. 3). Although the scope of the paper focuses on extending UML for quantum software, it could be explored how to represent in UML other fundamental properties of quantum computers (e.g., superposition, entanglement, etc.).

The proposal made in this paper is an attempt to extend UML to adapt it for hybrid information systems. When extending UML, we first should remember that UML is defined on an MOF (Meta-Object Facility) [51], which is a meta-metamodel. This therefore means that UML is a metamodel which is used to define different UML models, and the extension of UML consequently consists of extending the UML

metamodel. It is necessary to bear in mind that all the metamodel definitions have both an abstract syntax (that describes the concepts, their characteristics, and interrelationships in the metamodel language) and a concrete syntax (that defines the specific textual or graphical notations required for the abstract elements).

Taking this into account, it is possible to extend UML by following any one of three different approaches [18, 52].

- *A new instance of the MOF model* This approach consists of creating a completely new metamodel based on MOF. The result of this heavyweight extension is a new Domain-Specific Modelling Language (DSML).
- *Derivation of a new UML metamodel* This approach adds new metamodel elements to the existing one. As occurs with the first approach, it is a different metamodel, but at least it considers the original UML metamodel as is.
- *UML profile* This is a lightweight extension approach that is based on the UML built-in extension mechanism: UML Profiling. UML profiles are created as a set of stereotypes, tagged values and constraints defined for some of the existing UML elements, which changes or adds semantics, so in practice we can consider stereotyped elements as different ones.

On the one hand, it is clear that the first two approaches have a powerful expressiveness, since conformity with UML is not required (particularly in the case of new MOF instances). On the other hand, the expressiveness of the UML profiles is limited. However, standardisation and conformance are better for UML profiles, since the extension is fully compliant with UML. This advantage is a key aspect as regards the use of the defined profile with existing UML modelling tools. Consequently, we believe this approach will serve to maximise the adoption of the UML extension by industry.

Moreover, if we consider the future evolution of the UML extension, it is easier to maintain extensions that have been defined as UML profiles since the associated modelling tools do not need to be adjusted after each change, as occurs with a DSML. In fact, DSMLs (approaches 1 and 2) usually end up with an overloaded and imprecise language. These advantages lead us to believe that the UML profile is the best option to define the UML extension for hybrid information systems.

The quantum UML profile is defined as a set of stereotypes and existing UML elements to which such stereotypes apply. A UML profile is defined as a package containing a set of defined stereotypes (that may or may not have a specific image). The UML profile must then be applied to a certain model. The particular attributes used to filter which UML elements are available when the profile is applied are *metamodelReference* and *metaclassReference*. When a stereotype is applied to a model element, the values of the properties are traditionally referred to as tagged values.

The quantum UML profile does not address all the existing UML elements nor the whole set of UML diagrams. It focuses instead on the particular diagrams that make sense for the design of hybrid information systems. While a UML model consists of elements such as packages, classes, and associations, several UML diagrams may be built under the same UML model. UML diagrams add graphical representations of the parts of a UML model, e.g., nodes connected by paths. UML considers fourteen different kinds of diagrams divided into structure and behaviour diagrams. The proposed

quantum UML profile covers the use cases, class, sequence, activity, and deployment diagrams.

For each kind of quantum UML diagram, explained in the following sections, we provide two figures. First, the excerpt of the UML profile that supports that kind of diagram. Second, a UML diagram of the respective type that serves as an example to illustrate the use of the UML profile. For the UML profile metamodel of each diagram, we present at the left-hand side the part of the UML metamodel as is, i.e., the existing metaclass elements. Light grey elements are the non-abstract entities and are, therefore, the elements available to be included in the UML diagram. In the UML profile metamodel of each diagram, we provide in the right-hand side the UML profile elements proposed plus the leftwards arrows from stereotypes to the existing metaclass elements, which indicate that the properties of a metaclass are extended using the respective quantum stereotype.

The whole UML Profile has been defined with Papyrus, an industrial-grade open-source tool that allow to define and manage UML profiles following the MDE principles. The Papyrus project files for the UML profile, including model images with high resolution, can be accessed at [53].

4.2 Use case diagrams

A use case diagram is a type of behaviour diagram for defining what the information system is supposed to do, i.e., the requirement modelling [16]. The key concepts used in this diagram are actor, use case, and subjects (any classifier) (see Fig. 1). Each subject represents a system under consideration to which the use case applies. Use cases can be related with each other through two relationships: include and extend (see Fig. 1).

For hybrid information systems we propose five stereotypes in the quantum UML profile (see Fig. 1) that are capable of meeting the following modelling challenges in classical-quantum software:

- The definition of quantum environments where a given quantum functionality is executed. Although the hybrid information systems must be seen as a whole, a distinction should be made as to those use cases where the functionality performed is based on quantum software. For example, some uses cases might be stereotyped with ≪Quantum≫. It should be noticed that use cases are employed at the analysis stage and their purpose is not to design the solution but simply to depict the problem domain. Hence, in most of the enterprise, management information systems, it does not make sense to distinguish classical and quantum use cases in this point. Nevertheless, there are other systems in which specific use cases (e.g., those related to quantum simulations in physics or quantum chemistry) that automatically constrain the usage of quantum software. As a result, if this decision is already known at the analysis stage, this information should be represented since it will be valuable for the design stage. The challenge of determining quantum use cases is addressed in [40].
- The quantum environments that represent the technology stack (software and hardware) where a specific use case is performed should be denoted as an actor element

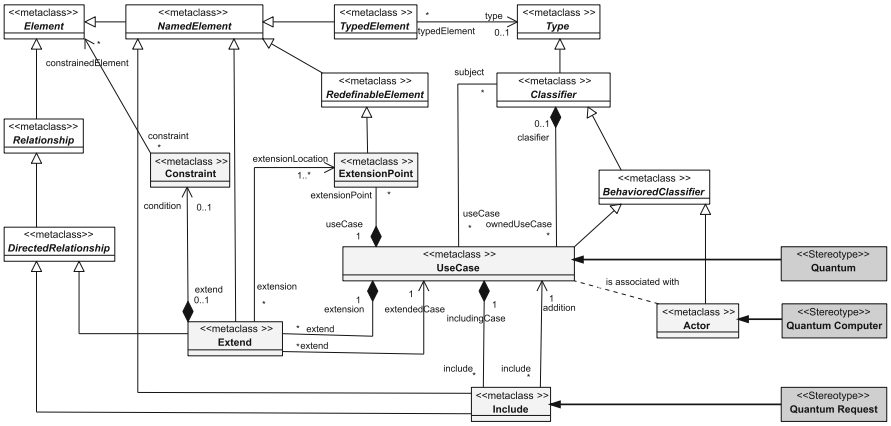

Fig. 1 Quantum UML profile for use case diagrams

with the ≪Quantum Computer≫ stereotype. These actors should be connected through associations with the ≪Quantum≫ use cases.

- The definition of quantum requests from ordinary use cases to quantum-based use cases could be denoted as well. As explained before, this depends on the prior knowledge about exactly what use cases are ≪Quantum≫. The proposal considers the stereotype ≪Quantum Request≫ that can be applied under include relationships. Although this information is not crucial at the analysis stage, it can help to trace such quantum requests in the following design diagrams. The semantics of the ≪Quantum Request≫ stereotype is a call to any component encoded as quantum software, which must run on a quantum device. This does not mean that the request itself is related to anything inherently quantum, beyond the fact that the response to that request could be quantum and thus need some kind of transformation.

Figure 2 provides an example of a use case diagram that employs the proposed stereotypes. The left-hand side of the diagram represents the use cases of the classical information subsystem with their common relationships (include and extend) as well as various actors. In the classical software part, there could be some functionality that requires the accomplishment of quantum algorithms. The use case shown in the example is defined without any stereotype but defines an include relationship with another ≪Quantum≫ use case. The right-hand side of Fig. 2 defines the ≪Quantum≫ use cases based on (or supported by) quantum software. These kinds of use cases should never be directly related to human actors, since they are typically performed from classical use cases that work as a driver. Nonetheless, the ≪Quantum≫ use cases are related with actors that represent quantum environments stereotyped with ≪Quantum Computer≫ (see Fig. 2). In this example, there is only one quantum environment. However, various quantum computers could be considered for the modelling of complex hybrid information systems. For instance, let us imagine a scenario in which some quantum-based use cases are supposed to be developed in a quantum environ-

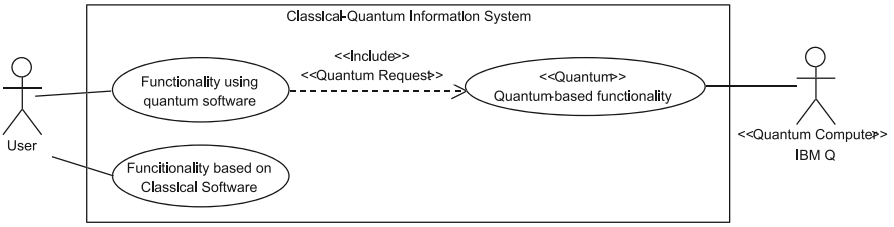

Fig. 2 Example of a use case diagram for a hybrid information system

ment based on quantum gates, while another optimisation-based functionality is to be supported in quantum annealing systems.

4.3 Class diagrams

Figure 3 summarises the UML metamodel part for class diagram elements as well as for those stereotypes defined in the quantum UML profile that can be applied to certain elements. In designing hybrid information systems, a class diagram could be used for modelling classical software (as has been the case up to now), while for quantum software, class diagrams could be used with the following additional purposes:

- First, the design of quantum software components is modelled with classes and packages that will support functionality and allow it to be developed as quantum circuits, algorithms, or similar artifacts. For this purpose, the ≪Quantum≫ stereotype can be used with both whole packages and individual classes.
- The definition of the class drivers that manage invocations to the quantum software components. This is represented by a class element with the stereotype ≪Quantum Driver≫.
- Design of quantum calls or requests from the ≪Quantum Driver≫ classes in the classical software packages to ≪Quantum≫ classes. Modelling these calls is optional and we therefore propose the stereotype ≪Quantum Request≫ which can be used with both association classes and dependencies. First, the use with an association class between the ≪Quantum Driver≫ class and the ≪Quantum≫ class allows one to link cost and optimisation functions to the quantum request. Second, a ≪Quantum Request≫ dependency represents the same call but in a more simplistic way, i.e., without further information. Moreover, since the quantum request is triggered from an operation in the class, the ≪Quantum Request≫ stereotype can be also used with operation elements to distinguish it from other auxiliary operations in the driver class.

Figure 4 shows an example of a class diagram that represents the architecture of a hybrid information system. The left-hand side provides the traditional three-layer architecture for the classical part, i.e., presentation, business logic, and persistency packages. The right-hand side shows the classical-quantum part. The classical-quantum logic package contains the quantum driver that performs quantum requests to different quantum algorithms which are placed in a package named quantum logic.

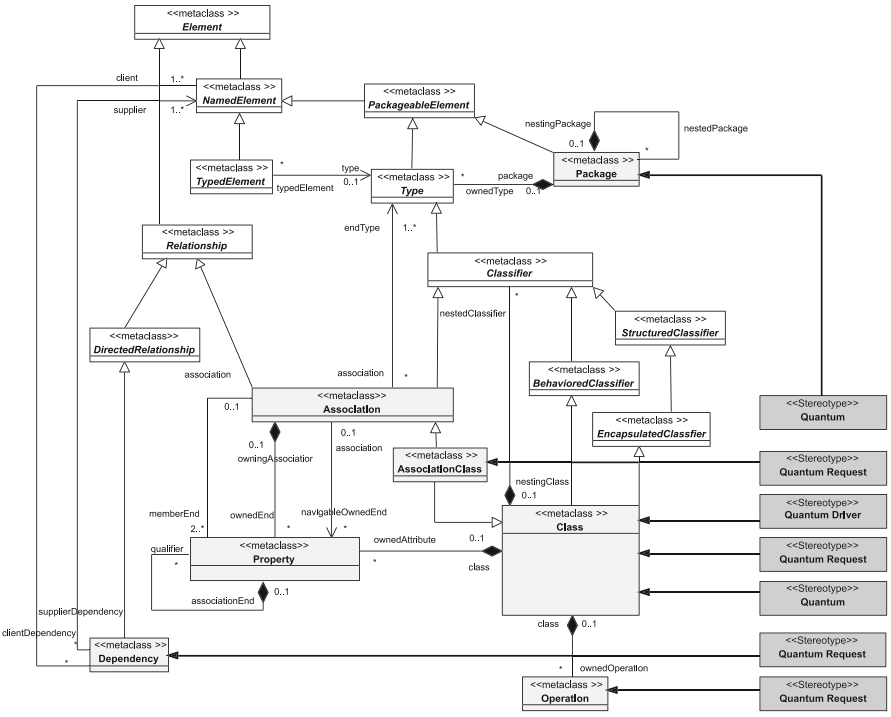

Fig. 3 Quantum UML profile for class diagrams

In the example of Fig. 4, the quantum request is modelled with an association class that is associated with the cost and optimisation functions. This represents only one possible design pattern for designing requests in hybrid information systems and it is not mandatory to follow this example in all cases. For instance, the quantum requests could be designed with a dependency, with the ≪Quantum Request≫ stereotype, between classes or packages. Furthermore, in some designs, quantum requests might not be of interest and so could be omitted.

Regarding the quantum software components, the structure diagrams do not show the details of dynamic behaviour, which are instead represented by behavioural diagrams. Hence, quantum algorithms are simply represented, by black box elements, as classes. However, some behaviour diagrams may show relationships to the behaviours of the classifiers exhibited in the class diagrams. The respective quantum circuits can therefore be modelled with activity diagrams according to the proposed UML profile (cf. Sect. 4.5), which could be referenced in diagram overview elements plus a ≪refine≫ dependency to the ≪Quantum≫ class.

4.4 Sequence diagrams

Sequence diagrams are interaction UML diagrams that allow one to model the sequence of messages that are exchanged, along with their corresponding occurrence

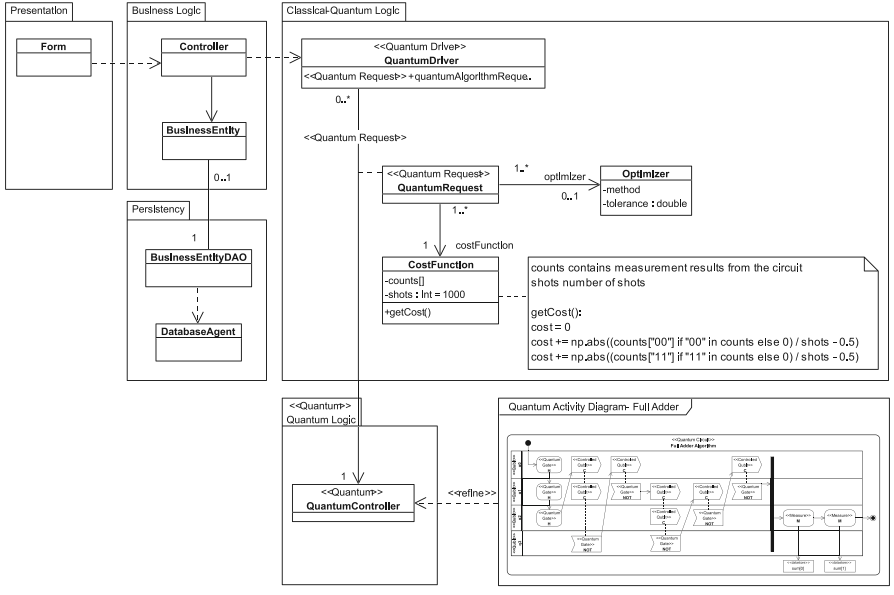

Fig. 4 Example of a class diagram for a hybrid information system using the quantum UML profile

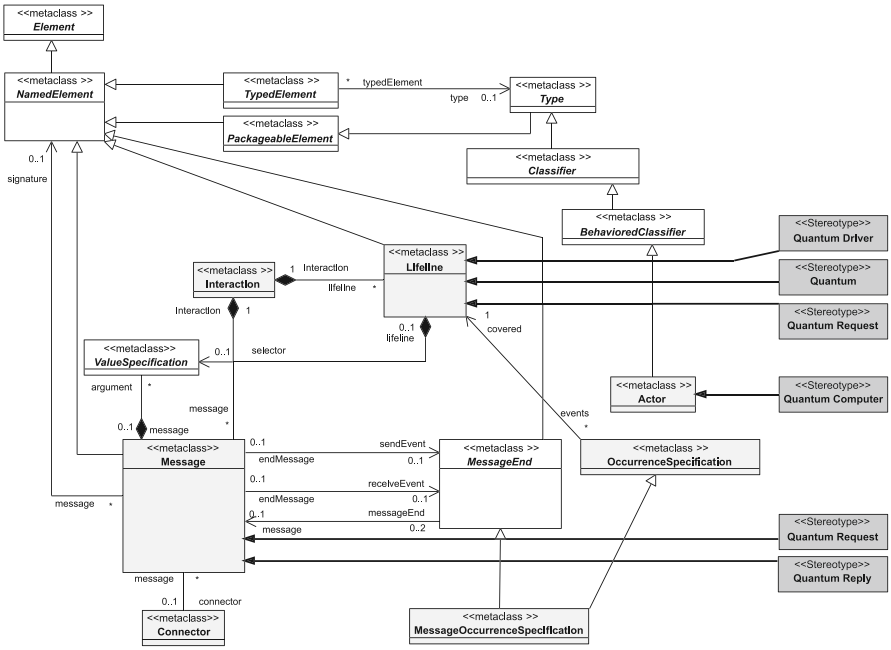

Fig. 5 Quantum UML profile for sequence diagrams

specifications on the lifelines (see Fig. 5). These diagrams simultaneously capture two aspects, the object and the time dimension. Sequence diagrams can be used in both analysis and design stages. Both kinds of sequence diagrams are useful for modelling some additional aspects during the development and modernisation of hybrid information systems. Those used in the analysis stage are related to the elements introduced by the quantum UML profile for use cases (cf. Sect. 4.2) for modelling use case scenarios. Additionally, design sequence diagrams are used for depicting dynamic behaviour regarding class diagrams (cf. Sect. 4.3). This section focuses on the design sequence diagrams.

Figure 6 shows an example of a sequence diagram that employs the quantum UML profile for the design of hybrid information systems. The first three lifelines to the left illustrate a common interaction between objects in the three-tier architecture previously defined in Fig. 4.

The same stereotypes defined for classes are also used for lifelines for the quantum software part (the right-hand side of Fig. 6). Thus, the ≪Quantum Driver≫ lifeline is responsible for creating a new instance object of ≪Quantum Request≫, which in turn creates instances of the classes *Optimizer* and *CostFunction* for managing specific requests to ≪Quantum≫ components. Since multiple quantum requests could be made to the quantum algorithm in accordance with the optimiser and cost function entities, a loop fragment may be added in that part of the diagram. It should be noticed that the ≪Quantum Request≫ stereotype is also used in this diagram to define what the messages are between classical and quantum software. Additionally, the ≪Quantum

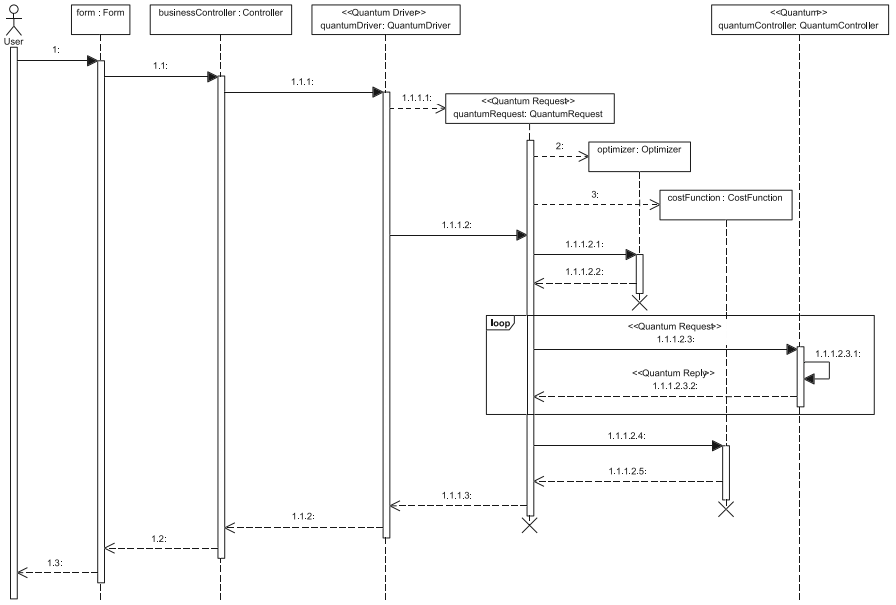

Fig. 6 Example of design sequence diagram for hybrid information systems

Reply≫ stereotype is used in this diagram to denote that the quantum algorithm answer has been sent by the ≪Quantum≫ component. After the loop of quantum request and reply, the cost function is applied to translate the quantum answers to a useful classical answer. This information is then sent in reply to the actor through all the involved lifelines (see Fig. 6).

4.5 Activity diagrams

Apart from their ordinary use for designing classical software, activity diagrams can be used for modelling quantum circuits, as was suggested in [45]. Ordinary activity diagrams can show the control flow between actions, i.e., these diagrams can depict concurrency, branch, control, and object flow, etc. (see Fig. 7).

Figure 8 provides an example of a UML activity diagram for a full adder algorithm based on [45]. Quantum algorithms are represented by a single compound activity with the stereotype ≪Quantum Circuit≫. The entire circuit is, therefore, defined in this activity, and the compound activity can be reused in other circuits, as often occurs in quantum programming. The various activity partitions (graphically represented as horizontal swim lanes) can be defined in the parent activity by employing the ≪Qubit≫ stereotype. The circuit has as many activity partitions as different qubits used in the algorithm. All the different quantum gates applied in the circuits are, therefore, represented as action elements and are placed in their respective swim lanes, according to the qubit under which the gate is applied or controlled. On the one hand, ordinary quantum gates (such as H, Y, Z, etc.) are represented as call operation actions plus the

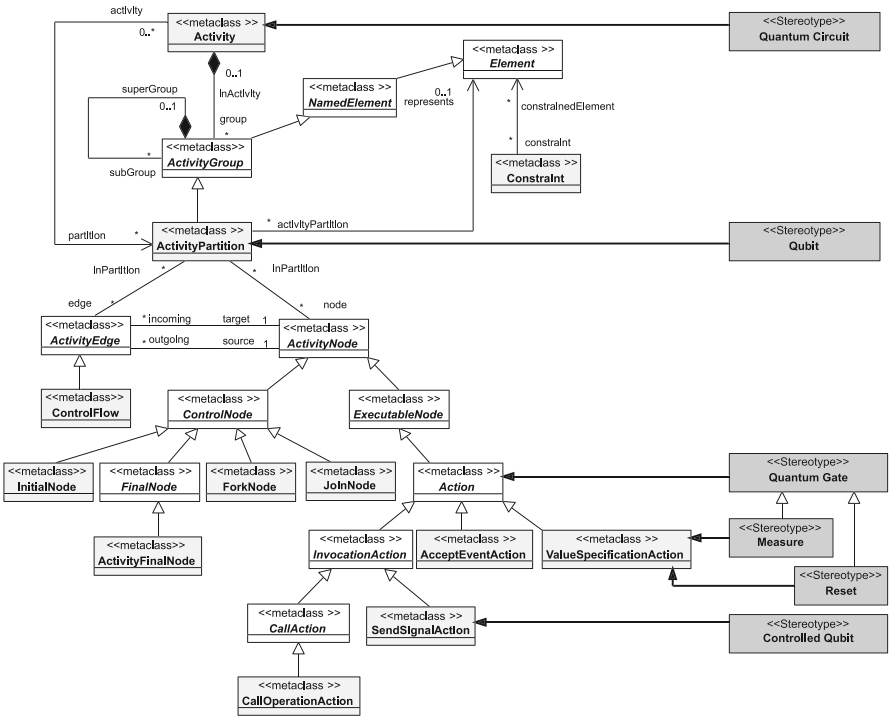

Fig. 7 Quantum UML profile for activity diagrams

≪Quantum Gate≫ stereotype. On the other hand, conditional gates are represented by multiple action elements. The control qubits are represented by send signal action elements with the stereotype ≪Controlled Qubit≫, while the gate applied is represented by the counterpart element, accept event action, plus the ≪Quantum Gate≫ stereotype (see Fig. 7). Additionally, to add the semantic concerning the relationships between the control qubits, various UML constraint elements are established between the various action elements involved [45]. Apart from these core elements, special operations, such as qubit measuring and qubit resetting, are represented by value specification action elements and their respective stereotypes, ≪Measure≫ and ≪Reset≫.

The quantum UML profile proposes the use of this kind of diagrams to represent quantum circuits in a similar way to the existing notations. On the one hand, it provides two benefits: the design of quantum circuits is UML-compliant (they can therefore be integrated with other UML elements and with whole diagrams) and this notation is supported for the existing UML modelling tools. On the other hand, the use of activity diagrams for this purpose i) can detract from the primary use of such UML diagrams; and ii) new quantum gates and specific notation cannot be supported by the activity diagrams' elements. As a result, we believe that the best long-term option is, at the next revision of the UML standard, to include a new kind of diagram to support the modelling of quantum circuits. Alternatively, there are other UML-based proposals

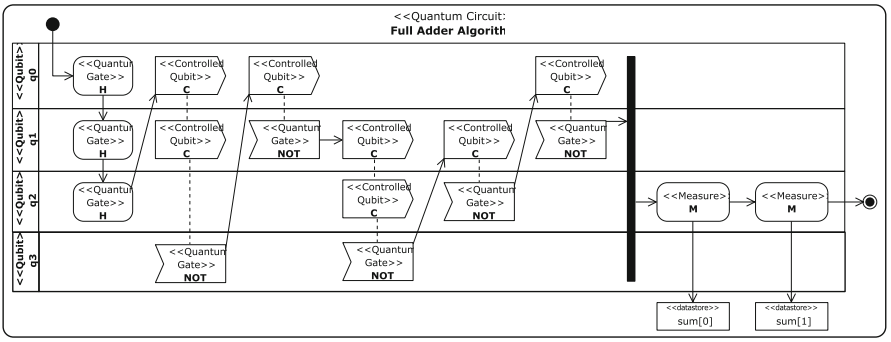

Fig. 8 Example of an activity diagram for a quantum full adder algorithm

for modelling the state superposition and probabilistic nature of quantum programs, for example by using state machines [44]. These proposals are also compatible with UML models using the proposed profile.

4.6 Deployment diagrams

A deployment diagram is a kind of structure diagram used in modelling the execution architecture of systems and the assignment of software artifacts to system elements (nodes). These diagrams capture relationships between logical and/or physical components of systems and the information technology artifacts assigned to them (see Fig. 9).

The purpose of deployment diagrams is to plan the architecture of a system and to document the deployment of software components. In the context of the design of hybrid information systems, deployment diagrams can be used with the following purposes:

- To show the structure of the run-time classical-quantum system.
- To capture the hardware that will be used to implement the system, i.e., the ≪Quantum≫ stereotype applied to different nodes, as well as the quantum components deployed inside.
- To act as a link between different nodes of the hardware, i.e., between the classical server acting as a driver and the quantum computer that executes the ≪Quantum≫ components.
- To model the communication paths between physical hardware elements. For example, deployment diagrams can capture the quantum requests made from classical to quantum software artifacts (see the ≪Quantum≫ stereotype in Fig. 9 as applied to the Artifact element).

Figure 10 shows an example of an archetype deployment diagram for hybrid information systems. The right-hand side defines the quantum side where the artifacts that correspond to the quantum algorithm are deployed in a ≪Quantum≫ node. It should be noted that specific interfaces between the Quantum Driver component and the Quantum Component can be modelled. These interfaces may be modelled as classes

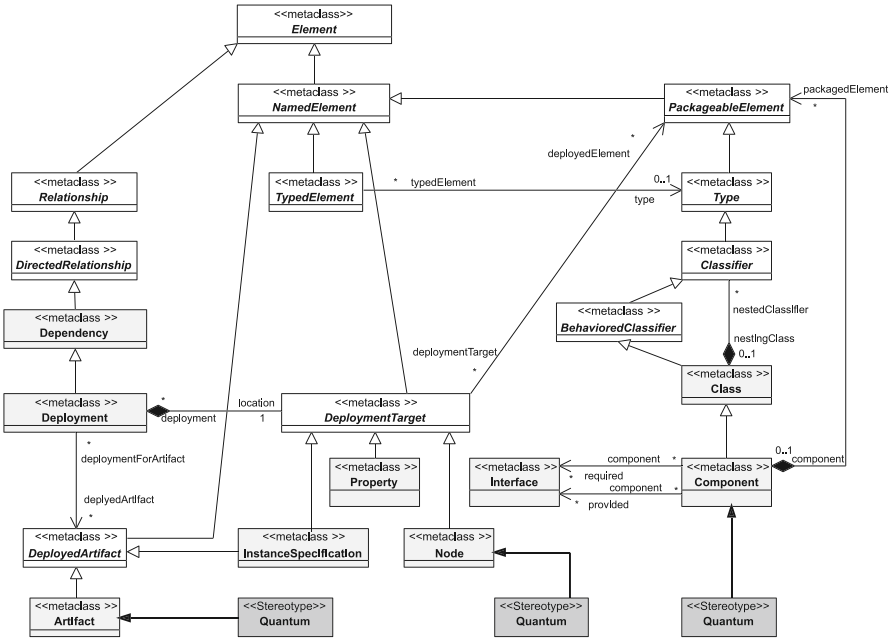

Fig. 9 Quantum UML profile for deployment diagrams

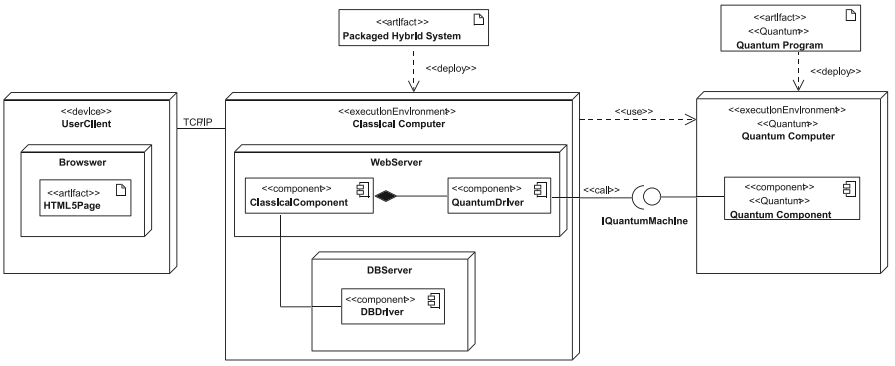

Fig. 10 Example of a deployment diagram for a hybrid information system

that can reuse the ≪Quantum Request≫ stereotype previously depicted (cf. Sect. 4.3). Apart from those stereotypes, the deployment diagram in Fig. 10 uses other ordinary stereotypes already defined in UML (e.g., ≪device≫, ≪executionEnvironment≫, and ≪call≫).

5 Discussion

The proposed quantum UML profile has been disseminated in accordance with the kind of UML diagrams (check the Papyrus project in [53]). However, most of the stereotypes included in the UML profile are applicable to various elements (see Table 1). For example, the ≪Quantum≫ stereotype is applied to seven (7) different elements that are involved in four (4) different diagrams (i.e., the stereotypes are associated with specific UML elements instead of types of diagrams). According to the UML standard [16], various elements can be used in different diagrams. Thereby, the same is true of the quantum UML profile, which implies that some stereotypes could be applied to additional diagrams in addition to those explained in this paper. For example, the package element (which could be stereotyped with ≪Quantum≫) could be used in most of the UML diagrams. Concerning the structure diagrams, object diagrams could employ all three (3) of the stereotypes that can be applied to the class element. In a similar manner, component diagrams can also use the ≪Quantum≫ stereotype applied to the component element. Concerning behaviour diagrams, another example is that of communication diagrams, which are in some way equivalent to the sequence diagrams, although the focus is on the structural organisation of the lifeline elements instead of the timeline definition. Communication diagrams can include all the stereotypes defined for the following elements: actor, lifeline, message and swimlane.

With regard to the usage of UML, research and practitioner communities are quite different. On the one hand, researchers usually focus on formal modelling notations for special-purpose solutions with a scope of depth over breadth. On the other hand, practitioners often prioritize practicality over rigor to find general-purpose solutions that can provide system architecture as a big picture in development projects, i.e., they focus on breadth over depth.

Having in mind this, research community could use the Quantum UML profile as a basis for establishing new research questions and developing related solutions. For example, as we stated before, UML provides different kinds of diagrams with which to look at information from varying perspectives and to represent the views of different concerns. Research community could wonder whether quantum systems require further perspectives (viewpoints) in addition to the ones provided by UML for specifying "classical" systems. Algorithmics are usually discarded in classical UML models since it contains low level details. However, this part is of paramount for quantum software. Quantum algorithmics must be defined under special rules, so this aspect may have a specific view in UML. Related to this, quantum circuits have been proposed to be modelled as UML activity diagrams. Nonetheless, one could think that this notation does not provide any kind of abstraction regarding existing no-UML notations. Still, we believe this allows to integrate quantum circuits' information with other UML elements in whole UML models. Alternatively, in the future, a new kind of UML diagrams could be proposed to represent this with the new quantum software elements: qubits, gates, etc. Other related question that can raise in the research community is if traditional software engineering methods work equally well for quantum systems. Maybe new software engineering methods are needed, and therefore new notations (as UML diagrams) will be needed too. In this regard, the design methodological counterpart for hybrid information systems should be investigated. Thus, each kind of

Table 1 Summary of the stereotypes defined in the UML profile and related elements

Stereotype	Use case				Class					Sequence	
	System	Use case	include	Actor	Package	Class	Association class	Dependency	Operation	Lifeline	Message
Quantum		x			x	x				x	
Quantum computer				x							
Quantum request			x			x	x	x	x	x	x
Quantum reply											x
Quantum driver						x				x	
Quantum algorithm											
Quantum circuit											
Qubit											
Quantum gate											
Measure											
Reset											
Controlled qubit											

Table 1 continued

Stereotype	Activity						Deployment		
	Swimlane	Activity	Activity partition	Action	Accept even action	Send signal action	Node	Component	Artifact
Quantum							x	x	x
Quantum computer									
Quantum request									
Quantum reply									
Quantum driver									
Quantum algorithm	x								
Quantum circuit		x							
Qubit			x						
Quantum gate				x	x				
Measure				x					
Reset				x					
Controlled qubit						x			

diagram considered in the quantum UML profile should be defined as the outcome of a set of activities or steps according to a software design methodology.

For practitioners, the main implication is that they can import the quantum UML profile in their modelling tools and can start to define the necessary analysis and design models for capturing the quantum aspects of the hybrid information systems. It also means that they do not need to learn a new modelling language or tool. Moreover, the effort required for analysing the problem domain and defining a conceptual architecture for the target hybrid information system can be still, in some ways, kept independent of the coding tasks from the UML models, which could be mainly addressed by other types of professionals.

6 Example of application

This section provides an example of use of the proposed quantum UML profile in order to analyse its applicability and suitability to design and build hybrid software systems. The target system is a financial web application to estimate the pricing call option payoff. The problem definition and its context are introduced in Sect. 6.1. Then, the analysis model through a use case diagram is presented in Sect. 6.2, while the design model through a class diagram is depicted in Sect. 6.3. Both models have been modelled through Papyrus tool and the whole modelling project is in [54]. Finally, Sect. 6.4 shows a functional web application that has been implemented by following the design, which is available in [55].

6.1 Problem definition and context

The example presented is inspired by an example provided in the QisKit finance library for estimate the payoff in pricing European call options [56]. In financial services, a 'call' is an option contract that gives the right, but not the obligation, to buy the underlying asset at a predetermined price before or at expiration day. European, against American model, can be exercised only at expiration. Having that financial model, the problem is to estimate the payoff, i.e., the option pricing and risk analysis. That problem is not trivial to be addressed by classical computers and has been already addressed through quantum amplitude estimation algorithms [57, 58].

The example provided by QisKit shows a plain interactive notebook where classical and quantum parts are merged, and many parameters and information are static. This is expected since that is an example focusing on demonstrating the application of quantum algorithms to resolve the financial problem. What we propose in our example is to design and develop a functional web application (as a hybrid software system) to be used by brokers and traders to estimate the payoff for call options. Thus, the web application could integrate some classical parts, e.g., to define the call options by means of some financial parameters, as well as to fix the objective like a pricing spot, etc. Thus, the application is able to estimate the payoff through the request to the quantum algorithm mentioned before. The system also supports the translation of the quantum algorithm's response to an answer understandable by the broker.

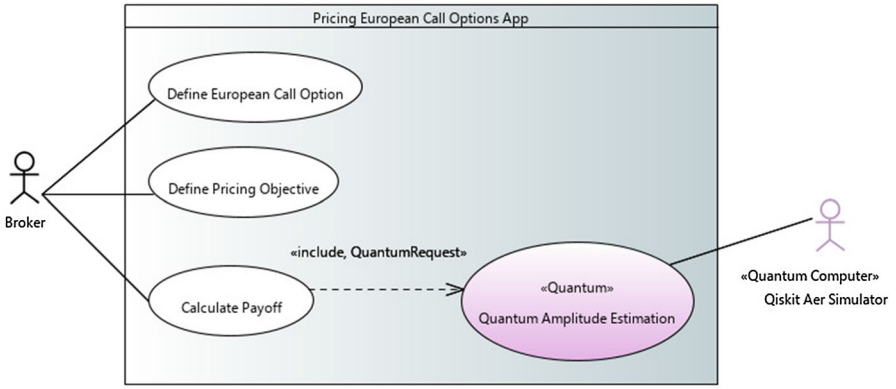

Fig. 11 Use case diagram with the quantum UML profile for a pricing European call options application

6.2 Analysis model

The analysis model consists of a case use diagram (see Fig. 11) to define the main usage scenarios of the hybrid software system. The main functionality of the system is estimation of the payoff for a given call option (see use case 'Calculate Payoff'). This use case includes (stereotyped with ≪QuantumRequest≫) another use case that performs a quantum algorithm (stereotyped with ≪Quantum≫) to estimate the payoff. Additionally, there are two use cases 'Define European Call Option' and 'Define Pricing Objective'. Those use case are respectively used for defining the financial model and the pricing parameters to compute the payoff. This system could include other classical functionalities related to the domain of the problem however it has been kept as simple as possible to be used as an example.

Finally, it should be noticed the actor in the right-hand side of Fig. 11 that represent the quantum computer used to execute the quantum algorithm. Due to the fact that this is an example of application it uses a built-in simulator of QisKit.

6.3 Design model

Considering the main functionality expressed in the use case diagram, a class diagram is modelled as the main artefact within the design model (see Fig. 12). This diagram follows the same architecture previously proposed in Sect. 4.3. First, the presentation package considers three forms that are directly related to the broker interaction with the system according to the three main use cases (see Fig. 11).

On the one hand, there are two forms related to the classical use cases, which have a dependency to the 'EuropeanController' in the 'businesslogic.controllers' package. This controller manages two entities, 'EuropeanCallProblem' and 'EuropeanCallObjective' that respectively represents information of the two cases studies associated with the financial model and the pricing parameters, i.e., the objective function.

On the other hand, the 'EvaluatePayoffForm' (associated with the quantum-based use case) has a dependency to the 'EuropeanCallDriver' (stereotyped with ≪Quantum

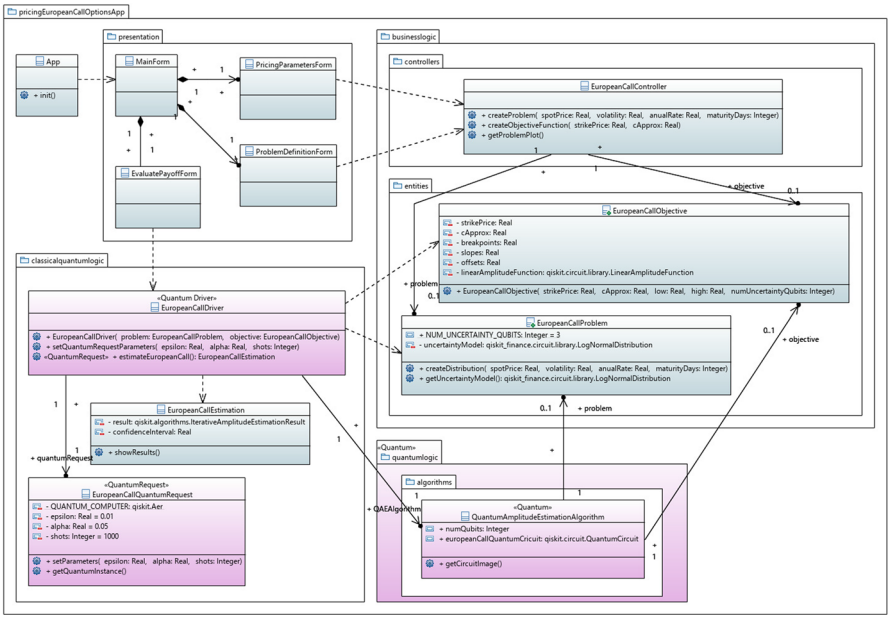

Fig. 12 Class diagram with the quantum UML profile for a pricing European call options application

Driver≫). That class is located in the 'classicalquantumlogic' package together with the classes 'EuropeanCallQuantumRequest' and 'EuropeanCallEstimation'. The first class is stereotyped with ≪QuantumRequest≫ and manages the information needed to perform the remote call to the target quantum computing (e.g., quantum computer endpoint, number of shots, and other parameters for the specific algorithm). Second, the 'EuropeanCallEstimation' class represents the cost function class that translate the reply of the quantum algorithm into a valuable/understandable classical answer, i.e., the payoff estimation in this example.

The 'EuropeanCallDriver' class has the attribute 'QAEAlgorithm', denoted as an association in the UML diagram with the class 'QuantumAmplitudeEstimationAlgorithm' (see Fig. 12). This class is located in the 'quantumlogic.algorithms' package and both are stereotyped with ≪Quantum≫. This stereotype points out that those software artefacts are coding or supporting some quantum algorithms. In particular, that class represents the code of the quantum amplitude estimation algorithm coded and presented in [56], which is based on Grover's algorithm. This class defines a problem and objective attributes with regards to the entity classes introduced before, since this quantum algorithm is defined generically and the outgoing algorithm is dynamically built according to those parameters [57, 58].

6.4 Implementation model

In order to validate the UML analysis and design models presented before, these have been used for coding a web application with the mentioned functionality. Hence, we

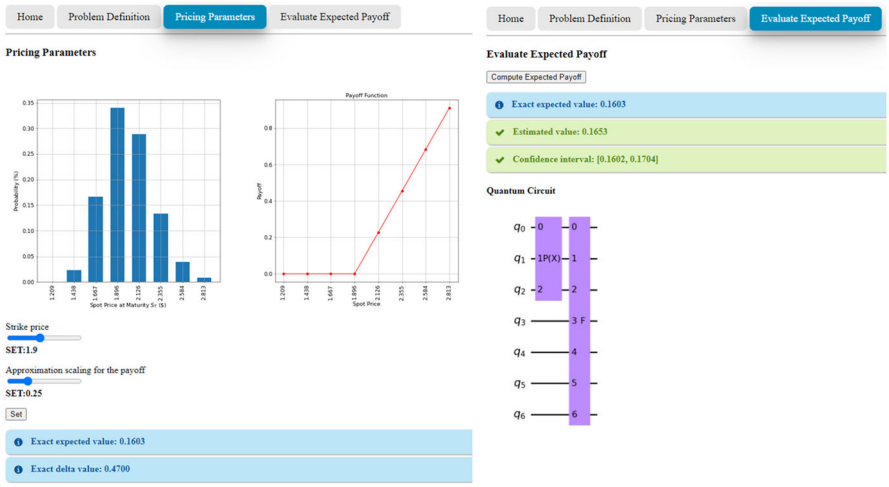

Fig. 13 Snapshots of the pricing European call options application [55]

used Python with QisKit and Flask as a quantum and web development frameworks respectively. The web application is available online in [55]. Figure 13 shows snapshots of two of the three main forms available. The left-hand side snapshot is the page in which the broker can define the pricing parameters, i.e., define the objective for a given call options problem. The right-hand side in Fig. 13 shows the results of the estimation of the payoff, as well as the resulting quantum algorithm.

7 Conclusions

This paper has proposed a quantum UML profile that attempts to address the lack of analysis and design techniques for hybrid information systems. The quantum UML profile consisted of a set of twelve stereotypes that can be applied in twenty standard UML elements. The use of the quantum UML profile has been demonstrated with at least five kinds of diagrams: use case, class, sequence, activity, and deployment. We believe that this UML profile is not only valuable for designing hybrid information systems, but also useful for the prior analysis undertaken by means of the use case and sequence diagrams. Additionally, the quantum UML profile can be employed for modelling the static (structure) and dynamic (behaviour) viewpoints of the hybrid information systems. The main implication is that the quantum UML profile, and the examples of its use with various UML diagrams, provides a set of design guidelines for the development of classical-quantum information systems. This proposal specifically suggests how the relationships between classical and quantum software should be managed, and how these relationships can be modelled in abstract designs.

The quantum UML profile provides a means for the design of hybrid information systems. However, nothing has been said about the methodological way to analyse and design such classical-quantum systems. Future research will need to be conducted in

this direction. Although there are already some methodologies based on UML for the modelling of classical information systems, for example the Rational Unified Process [59], specific methods need to be considered for designing hybrid information systems. Apart from the methodological part, another interesting point to be analysed in the future is how the outgoing UML models can be used by low-code tools that are able to automatically generate some parts of the code. Although such tools already exist for generating classical software, we believe that the generation of quantum code from UML models is an important concern and one that should be further investigated. Finally, a last point to be explored is how fundamental properties of quantum computing (apart from those at higher abstraction levels associated with quantum software) can be modelled in UML.

Acknowledgements This work is part of the SMOQUIN project (PID2019-104791RB-I00) and "QHealth: Quantum Pharmacogenomics Applied to Aging", 2020 CDTI Missions Programme, both funded by Spanish MICINN.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Egger DJ, Gambella C, Marecek J, McFaddin S, Mevissen M, Raymond R, Simonetto A, Woerner S, Yndurain E (2020) Quantum computing for finance: state-of-the-art and future prospects. IEEE Trans Quantum Eng 1:1–24
2. The Economist (2020) Wall street's latest shiny new thing: quantum computing. https://www.economist.com/finance-and-economics/2020/12/19/wall-streets-latest-shiny-new-thing-quantum-computing
3. Yudong C, Jonathan R, Olson Jonathan P, Matthias D, Johnson Peter D, Mária K, Kivlichan Ian D, Tim M, Borja P, Sawaya Nicolas PD, Sukin S, Libor V, Alán A-G (2019) Quantum chemistry in the age of quantum computing. Chem Rev 119(19):10856–10915
4. Cao Y, Romero J, Aspuru-Guzik A (2018) Potential of quantum computing for drug discovery. IBM J Res Dev 62(6):6:1-6:20
5. Njorbuenwu M, Swar B, Zavarsky P (2019) A survey on the impacts of quantum computers on information security. In: 2019 2nd International conference on data intelligence and security (ICDIS), pp 212–218
6. Christopher S (2021) How quantum computers could cut millions of miles from supply chains and transform logistics. https://www.forbes.com/sites/forbestechcouncil/2021/02/05/how-quantum-computers-could-cut-millions-of-miles-from-supply-chains-and-transform-logistics/
7. Zhang Y, Ni Q (2020) Recent advances in quantum machine learning. Quantum Eng 2(1):e34
8. Mueck L (2017) Quantum software. Nature 549(7671):171–171
9. Piattini M et al (2020) The Talavera manifesto for quantum software engineering and programming. In: Piattini M et al (eds) QANSWER 2020. QuANtum SoftWare Engineering and pRogramming, CEUR-WS, vol 2561, pp 1–5
10. Booch G (2018) The history of software engineering. IEEE Softw 35(5):108–114

11. Piattini M, Peterssen G, Serrano MA, Hevia JL, Pérez-Castillo R (2021) Towards a quantum software engineering. IT Prof 23(1):62–66
12. Rieffel EG, Polak WH (2011) Quantum computing: a gentle introduction (scientific and engineering computation). MIT Press, Cambridge
13. Wang SP, Sakk E (2021) Quantum algorithms: overviews, foundations, and speedups. In: 2021 IEEE 5th international conference on cryptography, security and privacy (CSP), pp 17–21
14. Pérez-Castillo R, Serrano MA, Piattini M (2021) Software modernization to embrace quantum technology. Adv Eng Softw 151:102933
15. Dey N, Ghosh M, Kundu SS, Chakrabarti A (2020) Qdlc—the quantum development life cycle. arxiv: abs/2010.08053v1
16. OMG. UML 2.5.1 The Object Management Group (2017). https://www.omg.org/spec/UML/2.5.1/PDF
17. Pérez-Delgado CA, Perez-Gonzalez HG (2020) Towards a quantum software modeling language. In: Proceedings of the IEEE/ACM 42nd international conference on software engineering workshops, ICSEW'20, New York, Association for Computing Machinery, pp 442–444
18. Ribo JM, Franch J (2002) A two-tiered methodology to extend the uml metamodel. Universitat Politècnica de Catalunya. https://upcommons.upc.edu/bitstream/handle/2117/97437/R02-52.pdf
19. Maslov D, Nam Y, Kim J (2019) An outlook for quantum computing [point of view]. Proc IEEE 107(1):5–10
20. Barbosa LS (2020) Software engineering for 'quantum advantage'. In: Proceedings of the IEEE/ACM 42nd international conference on software engineering workshops, ICSEW'20, Association for Computing Machinery, New York, pp 427–429
21. Aaronson S (2008) The limits of quantum. J Sci Am 298(3):62–69
22. Ferrari D, Cacciapuoti AS, Amoretti M, Caleffi M (2021) Compiler design for distributed quantum computing. IEEE Trans Quantum Eng 2:1–20
23. McClean JR, Romero J, Babbush R, Aspuru-Guzik A (2016) The theory of variational hybrid quantum-classical algorithms. J New J Phys 18(2):023023
24. McCaskey A, Dumitrescu E, Liakh D, Humble T (2018) Hybrid programming for near-term quantum computing systems. In: 2018 IEEE international conference on rebooting computing (ICRC), pp 1–12
25. Geller A (2020) Introducing quantum intermediate representation (QIR). https://devblogs.microsoft.com/qsharp/introducing-quantum-intermediate-representation-qir/
26. Carleton AD, Harper E, Robert JE, Klein MH, De Niz D, Desautels E, Goodenough JB, Holland C, Ozkaya I (2021) Schmidt Douglas architecting the future of software engineering: a national agenda for software engineering research and development. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=741193
27. Johnston ER, Harrigan N, Gimeno-Segovia M (2019) Programming quantum computers: essential algorithms and code samples. O'Reilly Media
28. Silva Vladimir Practical Quantum Computing for Developers (2018) Springer
29. Weder B, Barzen J, Leymann F, Salm M, Vietz D (2020) The quantum software lifecycle. In: Proceedings of the 1st ACM SIGSOFT international workshop on architectures and paradigms for engineering quantum software, APEQS 2020, Association for Computing Machinery, New York, pp 2–9
30. Zhao J (2020) Quantum software engineering: landscapes and horizons. arxiv: 2007.07047v1
31. van den Brink RFM, Phillipson F, Neumann NMP (2019) Vision on next level quantum software tooling. In: COMPUTATION TOOLS 2019: the tenth international conference on computational logics, Algebras, Programming, Tools, and Benchmarking, IARIA, pp 16–23
32. Wille R, Chattopadhyay A, Drechsler R (2016) From reversible logic to quantum circuits: logic design for an emerging technology. In: 2016 International conference on embedded computer systems: architectures, modeling and simulation (SAMOS), pp 268–274
33. Wang S, Wang Z, Li W, Fan L, Cui G, Wei Z, Yongjian G (2020) Quantum circuits design for evaluating transcendental functions based on a function-value binary expansion method. Quantum Inf Process 19(10):347
34. Wille R, Fowler A, Naveh Y (2018) Computer-aided design for quantum computation. In: 2018 IEEE/ACM international conference on computer-aided design (ICCAD), pp 1–6
35. Genç HH, Aydın S, Erdal H (2020) Design of virtual reality browser platform for programming of quantum computers via vr headsets. In: 2020 International congress on human-computer interaction, optimization and robotic applications (HORA), pp 1–5

36. Bandic M, Zarein H, Alarcon E, Almudever CG (2020) On structured design space exploration for mapping of quantum algorithms. In: 2020 XXXV conference on design of circuits and integrated systems (DCIS), pp 1–6
37. Zhou X, Li S, Feng Y (2020) Quantum circuit transformation based on simulated annealing and heuristic search. IEEE Trans Comput Aided Des Integr Circuits Syst 39(12):4683–4694
38. Thompson N, Steck J, Behrman E (2020) A non-algorithmic approach to programming quantum computers via machine learning. In: 2020 IEEE international conference on quantum computing and engineering (QCE), pp 63–71
39. Wecker D, Svore KM (2014) Liqui: a software design architecture and domain-specific language for quantum computing. arxiv: abs/1402.4467
40. Chancellor N, Cumming R, Thomas T (2020) Toward a standardized methodology for constructing quantum computing use cases. arxiv: 2006.05846v1
41. Weder B, Breitenbücher U, Leymann F, Wild K (2020) Integrating quantum computing into workflow modeling and execution. In: 2020 IEEE/ACM 13th international conference on utility and cloud computing (UCC), pp 279–291
42. Weder B (2021) Quantme-quantum4bpmn, UST-QuAntiL. https://github.com/UST-QuAntiL/QuantME-Quantum4BPMN
43. Exman I, Shmilovich AT (2021) Quantum software models: the density matrix for classical and quantum software systems design. arxiv: 2103.13755v1
44. Ali S, Yue T (2020) Modeling quantum programs: challenges, initial results, and research directions. In: Proceedings of the 1st ACM SIGSOFT international workshop on architectures and paradigms for engineering quantum software, APEQS 2020, Association for Computing Machinery, New York, pp 14–21
45. Pérez-Castillo R, Jiménez-Navajas L, Piattini M (2021) Modelling quantum circuits with uml. In: 2021 IEEE/ACM 2nd international workshop on quantum software engineering (Q-SE), pp 7–12
46. ISO/IEC/IEEE (2017) International standard-systems and software engineering-vocabulary. ISO/IEC/IEEE 24765:2017(E), pp 1–541
47. Ralph P, Wand Y (2009) A proposal for a formal definition of the design concept. In: Design requirements engineering: a ten-year perspective, Springer, Berlin, pp 103–136
48. Lange CFJ, Chaudron MRV, Muskens J (2006) In practice: Uml software architecture and design description. IEEE Softw 23(2):40–46
49. Gamma E (2002) Design patterns—ten years later, Springer, Berlin, pp 688–700
50. Gemeinhardt F, Garmendia A, Wimmer M (2021) Modelling quantum circuits with uml. In: Second international workshop on quantum software engineering (Q-SE 2021), IEEE Computer Society
51. OMG (2016) Meta object facility (MOF^{TM}) version 2.5.1, The Object Management Group. https://www.omg.org/spec/MOF/2.5.1/PDF
52. Medvidovic N, Rosenblum DS, Redmiles DF, Robbins JE (2002) Modeling software architectures in the unified modeling language. ACM Trans Softw Eng Methodol 11(1):2–57
53. Pérez-Castillo R (2021) Quantum uml profile repository. https://github.com/ricpdc/quml
54. Pérez-Castillo R (2022) Example of application of quantum uml profile—pricing european call options. https://github.com/ricpdc/quml-example
55. Alarcos RG (2022) Pricing European call options app. http://alarcosj.esi.uclm.es:8080/
56. QisKit (2021) Pricing European call options. https://qiskit.org/documentation/finance/tutorials/03_european_call_option_pricing.html
57. Woerner S, Egger DJ (2019) Quantum risk analysis. NPJ Quantum Inf 5(1):15
58. Stamatopoulos N, Egger DJ, Sun Y, Zoufal C, Iten R, Shen N, Woerner S (2020) Option pricing using quantum computers. Quantum Open J Quantum Sci 4:291
59. Jacobson I, Booch G, Rumbaugh J (1999) The unified software development process, 1st edn, Addison-Wesley