

Reverse Engineering of OpenQASM3 Quantum Programs to KDM Models

Luis Jiménez-Navajas¹^a, Ricardo Pérez-Castillo¹^b and Mario Piattini²^c

¹*aQuantum, Faculty of Social Sciences & IT, University of Castilla-La Mancha Talavera de la Reina, Spain*

²*aQuantum, Information Technology and Systems Institute, University of Castilla-La Mancha Talavera de la Reina, Spain*

Keywords: Reverse Engineering, Quantum Computing, OpenQASM3, KDM.

Abstract: The development of quantum computing is following a substantial growth. This leads us closer to the implementation of practical solutions based on quantum software that address problems that are not computable by classical software in a practical timeframe. Hence, some companies will need to adapt their development practices and, so, their information systems to take advantage of quantum computing. Unfortunately, there is still a lack of tools, frameworks, and processes to support the evolution of current systems towards the combination of the quantum and classical paradigms into information systems. Hence, this paper presents a reverse engineering technique to generate abstract models based on the Knowledge Discovery Metamodel (KDM) by analyzing quantum software written in OpenQASM3. The main implication is that KDM models represent, in a technology-agnostic way, the different components and interrelationships of quantum software. These models then can be used to restructure and redesign the target hybrid information system.


1 INTRODUCTION


Quantum computing is the result of the application of some counterintuitive principles of quantum mechanics to computer science (e.g., superposition or entanglement) in order to represent information as quantum states (Gyongyosi & Imre, 2019). It allows the performance of certain algorithms which are unattainable for today's supercomputers. The expectation for quantum computing is so high with plenty of expected applications, like in chemistry (McArdle, Endo, Aspuru-Guzik, Benjamin, & Yuan, 2020), artificial intelligence (Dunjko & Briegel, 2018) and, even, in communications (Imre & Balazs, 2005). Large companies have invested a lot of resources in quantum computing and their potential applications (Julian van Velzen, 2022; Wang, 2021), which are close to be effectively implemented.


Probably, not all the business operations will be adapted into the quantum paradigm. This is because the performance of the classical software for some business operations is still (and will keep)

appropriate. Thus, the new software systems will evolve toward hybrid software systems, i.e., classical and quantum software will operate together (Pérez-Castillo, Serrano, & Piattini, 2021). Therefore, companies that want to be benefited, will need to evolve toward hybrid information systems (Houekpetodji, Anquetil, Ducasse, Djareddir, & Sudich, 2021). In such hybrid systems, the classical part will control and receive all the outputs generated by the quantum algorithms executed generally in remote quantum computers in the cloud (Nguyen, Usman, & Buyya, 2022).

Throughout the years, a mass of knowledge and experience has been accumulated in the field of software engineering and, in particular, in software modernization (i.e., the progress of reengineering by applying a Model-Driven Engineering approach). Software modernization has allowed organizations to develop and evolve high-quality software in compliance with standards and following well-proven practices (Durelli et al., 2014). Thus, some of these software engineering techniques and methods

^a <https://orcid.org/0000-0001-6257-7153>

^b <https://orcid.org/0000-0002-9271-3184>

^c <https://orcid.org/0000-0002-7212-8279>

are being adapted for quantum software development (De Stefano, Pecorelli, Di Nucci, Palomba, & De Lucia, 2022; Li, Khomh, & Openja, 2021; Piattini et al., 2020). One of these processes is the Quantum Software Modernization process (Pérez-Castillo, Serrano, et al., 2021), where the process of software modernization considers the integration of quantum algorithms when designing the target hybrid software systems.

This paper takes place in the context of the Quantum Software Modernization process and, more specifically, in the reverse engineering phase. This phase is crucial since it is used to produce abstract representations of classical and quantum software. We are able to generate KDM (Knowledge Discovery Metamodel) models (Pérez-Castillo, De Guzman, Piattini, & Interfaces, 2011). As a part of the ADM initiative, the OMG released KDM within a broad set of proposed standards (Ulrich, 2010). KDM addresses the main challenges that appear in the modernization of legacy information systems (Pérez-Castillo, De Guzman, et al., 2011) and can contribute to the modernization of hybrid software systems. The usage of KDM allows to represent in a technology-agnostic way all the different components and interrelationships of quantum software.

This reverse engineering technique was firstly piloted for Q# programs (Jiménez-Navajas, Pérez-Castillo, & Piattini, 2020). The support for additional languages is mandatory, and in case of OpenQASM3 [13] (QASM hereinafter) is also crucial since it can be considered as de facto standard as it has been used as a base for developing many software frameworks and development environments (Garhwal, Ghorani, & Ahmad, 2021). Thus, this paper proposes a reverse engineering technique to analyze quantum programs developed in QASM and represent that information in KDM models. If we want to facilitate the evolution from/toward hybrid software systems, companies or organizations should not face restrictions on which quantum programs they can incorporate into their hybrid software systems. Having achieved agnostic representations of quantum software, it can be integrated along with classical software, which is a step forward in the modelling and designing of hybrid software systems.

The remain of this paper is structured as follows: Section 2 relates the actual state of Quantum Software Modernization and how the representation of quantum programs in KDM has been accomplished. Then, Section 3 explains how OpenQASM3 programs have been analyzed and modelled in KDM. Section 4 shows a preliminary evaluation of the

reverse engineering. Finally, Section 5 presents the conclusions and future work of this research.

2 BACKGROUND

This section is divided in two subsections. Section 2.1 describes the actual state of the Quantum Software Modernization and Section 2.2 shows the extension of KDM for quantum software.

2.1 Quantum Software Modernization

In today's world companies invest considerable amounts of effort in keeping their information systems competitive. This often implies performing maintenance activities in their systems while the preservation of its business knowledge is required (Paradauskas, Laurikaitis, & control, 2006). However, in some cases, this maintenance becomes difficult since the technology on which it was built has become obsolete. To address these problems during evolution of information systems, the Software Modernization process was proposed (Pérez-Castillo, de Guzmán, & Piattini, 2011). This process was developed by following the traditional reengineering approach alongside the Model Driven Engineering (MDE) principles. This revisited approach solves some of the shortcomings of traditional reengineering (the lack of formalization and standardization (Kazman, Woods, & Carrière, 1998)).

With the advent of quantum computing, many information systems could become obsolete, and companies will have to modernize them. This does not mean discarding the whole system, but first assessing which components of the system could/should be supported by quantum software and be modernized accordingly. Figure 1 presents the Quantum Software Modernization process as proposed in (Jiménez-Navajas et al., 2020), which advocates the use of standards widely accepted in the industry such as KDM and UML.

Quantum Software Modernization consists of three phases: reverse engineering, restructuring and forward engineering. In the reverse engineering phase (left-hand side of Figure 1) the different components of the current classical information system and quantum programs are represented in KDM models in a technology-agnostic way. This phase is the scope of this paper. In the restructuring phase (top center of Figure 1) the previously generated KDM models are transformed into high-level abstraction models and the target hybrid information system is designed.

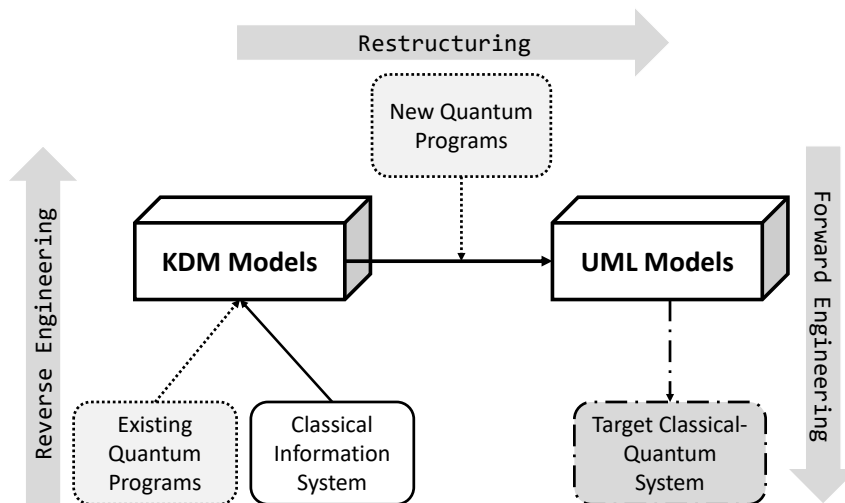


Figure 1: Quantum Software Modernization approach.

Finally, in the forward engineering phase (right-hand side of Figure 1), tools are used to automatically generate the code of the hybrid information system designed in the previous phase.

2.2 Quantum KDM Extension

To accomplish the task of modernizing legacy systems, the OMG proposed the KDM specification (ISO/IEC, 2009). This standard allows the complete representation of legacy information systems at a high level of abstraction while preserving all business knowledge.

Moving to the context of quantum computing, to represent in KDM models the different components that are used in quantum software, the metamodel must be extended. The extension of the metamodel has been carried out using the mechanism provided by the standard itself, i.e., by creating an "extension family". This extension family contains the different components that can appear in a quantum program (see Figure 2).

```
<extensionFamily id="id.0" name="quantum extension">
  <stereotype name="quantum programming language" id="id.1"/>
  <stereotype name="quantum program" id="id.2"/>
  <stereotype name="quantum operation" id="id.3"/>
  <stereotype name="quantum gate" id="id.4"/>
  <stereotype name="qubit" id="id.5"/>
  <stereotype name="qubit measure" id="id.6"/>
  <stereotype name="control qubit" id="id.7"/>
  <stereotype name="qubit array" id="id.8"/>
</extensionFamily>
```

Figure 2: KDM extension family.

Whenever a KDM model is generated from an OpenQASM3 program, the extension family is included in the model. Afterwards, when quantum

elements are represented, they have an attribute that references to a specific stereotype.

3 OpenQASM3 TO KDM

To carry out the generation of KDM models from QASM quantum programs, the same steps have been followed as in (Jiménez-Navajas et al., 2020). This procedure consists of creating two modules: one that analyses the code and extracts the necessary information, and another one that translates this information into the KDM standard. This new development has not been carried out in a standalone tool but integrated into QRev (Pérez-Castillo, Jiménez-Navajas, & Piattini, 2021). Section 3.1 will discuss how quantum programs are analyzed and Section 3.2 shows how the information extracted from the previous step is represented through KDM standard.

3.1 OpenQASM3 Parser

To extract information from QASM programs, a parser has been developed. This parser has been developed by means of ANTLRv4, a tool for generating parsers based on formal grammar definitions (Parr). Programs based on ANTLR contain the formal definition of the syntax and grammar of the language to be recognized. Using that grammar ANTLR generates the code of the language recognition tool, i.e., the parser. That parser is able to recognize the target programming language and generates an Abstract Syntactic Tree (AST) which represents the structure of the analyzed program. The

grammar employed was not defined from scratch, as an official QASM grammar is available in (Andrew W. Cross, 2020).

In order to make the explanation of the proposal presented as descriptive as possible, a running example of the project will be carried out using the quantum teleportation algorithm as an input program (see Figure 3).

```

1 OPENQASM 3;
2 include "stdgates.inc";
3 qubit[3] q;
4 bit crz;
5 bit crx;
6 h q[1];
7 cx q[1], q[2];
8 cx q[0], q[1];
9 h q[0];
10 crz = measure q[0];
11 crx = measure q[1];
12 if(crz==1) z q[2];
13 if(crx==1) x q[2];
    
```

Figure 3: Quantum teleportation algorithm implemented in OpenQASM3.

Quantum teleportation (Bouwmeester et al., 1997) is one of the most important algorithms in quantum computing as it allows us to move the state of one qubit to another without violating the no-cloning theorem (Wootters & Zurek, 2009).

Figure 4 shows the outgoing AST generated for the teleportation example showed before. Due to the space limitation, the example only presents a partial tree for lines 6 and 7. The nodes in the AST correspond with the same elements used in the

grammar rules defined for ANTLR. The AST elements and its relationships will be later considered to generate the KDM model.

3.2 KDM Generator

As mentioned before, the QASM parser and its corresponding KDM Generator are an extension of QRev. For the integration of these new parts in the tool, a complete refactoring of the code has been performed and the parser and the generator have been integrated following the ‘state’ design pattern. The state pattern is intended when the desired program is desired to “allows an object to alter its behavior when its internal state changes. The object will appear to change its class” (Gamma, Helm, Johnson, Johnson, & Vlissides, 1995). This pattern was chosen because, when receiving a quantum program as input, the file will behave as an object whose state is defined once its extension is known (‘.qs’ in the case of Q# programs and ‘.qasm’ if it is QASM). Depending on this, the KDM generator considers the specific transformations rules.

Once the AST has been generated, the module oriented to the generation of the KDM models goes through the tree and reads the nodes and some relationships in the tree. Only the relevant information from the nodes is extracted to make the KDM models as accurate as possible. Table 1 shows the name of the nodes of the AST of quantum elements and their representation in KDM. Each of the representations in KDM is in accordance with the definition of the quantum elements themselves.

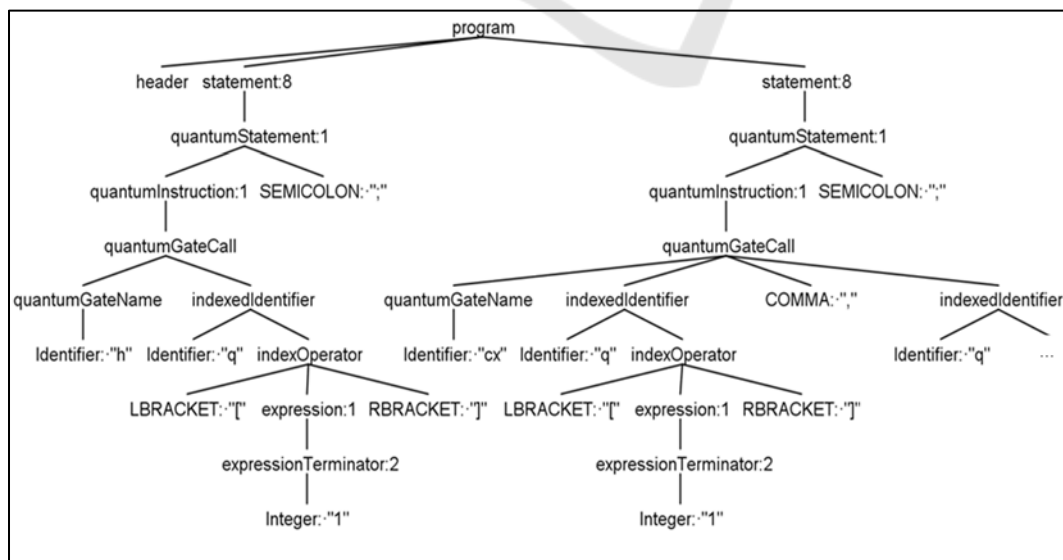


Figure 4: Fragment of the AST generated from the teleportation algorithm.

Table 1: Quantum elements and its AST and KDM representation.

Quantum element	AST modelling	KDM modelling	Stereotype
Quantum program	program	Compilation Unit	<ul style="list-style-type: none"> quantum program
Qubit	Quantum statement	StorableUnit	<ul style="list-style-type: none"> qubit control qubit qubit array
Quantum gate	Quantum instruction	Action Element	<ul style="list-style-type: none"> quantum gate qubit measure

In Figure 5 we can observe the actual KDM representation of a Hadamard gate with its corresponding "ActionElement" type and its stereotype pointing to `<<quantum gate>>` (as defined in the extension family). This KDM elements has six children, which respectively represent:

- The gate snippet (line 4).
- On which qubit is being applied (line 5-6).
- Being a qubit register, it specifies which qubit register is being applied (line 7-8).
- The readout of the qubit register (line 9-10).
- The writing of the qubit register (line 11-12).
- The flow of the information to the next quantum gate (line 12-13).

```

1 <codeElement id="id.20" xmi:type="action:
2 ActionElement" kind="operator" name="Hadamard"
3 stereotype="id.4">
4 <source language="OpenQASM3" snippet="h q[1]"/>
5 <actionRelation id="id.21"xmi:type="action:
6 Addresses" from="id.20" to="id.17"/>
7 <codeElement id="id.22" name="1"xmi:type="code:
8 Value" stereotype="id.5"/>
9 <actionRelation id="id.23" xmi:type="action:
10 Reads" from="id.20" to="id.16"/>
11 <actionRelation id="id.24"xmi:type="action:
12 Writes" from="id.20" to="id.17"/>
13 <actionRelation id="id.25"xmi:type="action:
14 Flow" from="id.20" to="id.26"/>
15 </codeElement>

```

Figure 5: KDM resulting from the analysis of a Hadamard gate.

Following the running example started in the previous section, in Figure 6 can be seen the teleport algorithm KDM's representation through the Eclipse model view. At the top of the model is located the extension family explained previously (cf. Section 2.2). Then, we have the declaration of the quantum program through the type "CompilationUnit" and through the attribute with the name of the program

(teleport.qasm). Subsequently, we can see the declaration of a *qubit* and other variables where the results of the quantum operations will be stored. These variables are represented in KDM by means of the type "StorableUnit", but the qubit has an attribute "stereotype" which points to the id of the element "qubit" in the extension family.

```

v ♦ Segment teleport_1647862584915.xml
v ♦ Extension Family quantum extension
  ♦ Stereotype quantum programming language
  ♦ Stereotype quantum program
  ♦ Stereotype quantum operation
  ♦ Stereotype quantum gate
  ♦ Stereotype qubit
  ♦ Stereotype qubit measure
  ♦ Stereotype control qubit
  ♦ Stereotype qubit array
v ♦ Model teleport_1647862584915.xml
v ♦ Compilation Unit teleport.qasm
  > ♦ Language Unit OpenQASM3 Common definitions
    ♦ Storable Unit q[3]
    ♦ Storable Unit crz
    ♦ Storable Unit crx
  > ♦ Element Hadamard
  > ♦ Element Controlled Not
  > ♦ Element Controlled Not
  > ♦ Element Hadamard
  > ♦ Element Pauli Z
  > ♦ Element Pauli X (Not)

```

Figure 6: Resulting KDM of the teleport algorithm.

4 PRELIMINARY EVALUATION

This section will consist of a preliminary evaluation of this tool extension. A total of 13 QASM programs extracted from the Qiskit example repository on Github (Qiskit, 2022) have been used to carry out this evaluation. Currently, there is not a large number of independent projects publishing quantum algorithms or programs. As a result, most of the resources or samples are usually found in the repositories of the programming languages (i.e., most of the examples developed in OpenQASM3 are found in their own repository), which in turn means that these same programs are the most visualized or used. Although there are 23 examples in the repository, in the end we filtered in 13 of them. Some programs were discarded since these contained sets of language declarations (like quantum gates' definitions) or these presented oracle's implementations (which are out of the scope of this project).

The results obtained from the evaluation can be observed in Table 2, where the name of the analyzed file and the variables used for its evaluation can be found, such as the element size, which consists of the

sum of qubits and quantum gates (i.e., *StorableUnits* and *ActionElements* in the KDM model), and the number of relationships (i.e., *Reads*, *Addresses* and *Flow* in KDM). In addition, it includes the time (in milliseconds) spent on generating the KDM. The execution environment consisted of a laptop with an i7 10510U with 2.30 GHz, 16 GB of RAM.

For this preliminary evaluation the precision, recall and f-measure of the KDM models generated have been calculated. To assess these values, we have employed the data that can be seen in the columns which concerns to the size of the elements, relationships, and missing elements. Missing elements are mainly composed of three structures: functions (and therefore all quantum gates that are implemented within a function), oracles, and measurement gates. Those missing elements are not represented in KDM, causing the comparison to be inaccurate.

Each missing quantum gate (either because it is defined in an oracle or implemented within a function) has been multiplied by 4 since at least its KDM representation would have a child “*Addresses*” and another “*Writes*”, but we considered convenient to compensate for the possible “*Flow*” and the “*Value*” that other quantum gates may have. Oracles that work as identity gates have not been considered as missing elements although are not represented.

About irrelevant elements (i.e., false positives) the reverse engineering technique does not retrieve anything undesired in the KDM model. Thus, the precision of the tool is always 100%.

Observing results in Table, there are specific programs such as 'msd' or 'rus' where the recall drops to almost 30%. As explained above, quantum gates within functions and oracles are not modelled in KDM yet, which causes the average recall obtained in programs such as those mentioned to drop. Nevertheless, the effectiveness of the tool could be considered acceptable since the study reported an average of 72.2% recall. Moving to the f-measure value, obtaining such a high percentage is not surprising as this measure has a direct relationship with the values obtained from recall and precision. However, this measure indicates that the generated models present a correct accuracy (81.2%).

In summary, reverse engineering of quantum programs allows to represent them in abstract models with an average recall of 72.2%, which proves that quantum programs could be integrated into the high-level design of hybrid information systems. The modelling of these new systems has yet to be formalized, but the state of the art (Azeem Akbar, Rafi, & Khan, 2022; Pérez-Delgado & Perez-Gonzalez, 2020; Weder, Barzen, Leymann, Salm, & Vietz, 2020).

Table 2: Results obtained from the preliminary evaluation.

File name	#Storable Units	#Action Elements	#Reads	#Addresses	#Flow	#Missing Elements	Precision (%)	Recall (%)	F-Measure (%)	KDM Generation time (ms)
adder	5	6	7	6	5	23	100	55.8	71.6	94
alignment	1	2	3	2	1	8	100	52.9	69.2	12
gateteleport	3	3	4	3	2	1	100	93.8	96.8	60
inverseqft1	2	17	17	17	16	4	100	94.5	97.2	77
inverseqft2	5	12	12	12	11	4	100	92.9	96.3	13
ipe	2	4	0	4	3	12	100	52.0	68.4	14
msd	4	6	8	6	5	75	100	27.9	43.6	88
qec	4	6	6	6	5	17	100	61.4	76.1	8
qft	2	13	19	13	12	0	100	100.0	100.0	15
qpt	2	2	0	2	1	1	100	87.5	93.3	8
rb	2	8	10	8	7	1	100	97.2	98.6	4
rus	4	4	0	4	3	31	100	32.6	49.2	13
teleport	4	8	10	8	7	4	100	90.2	94.9	13
Min	1.0	2.0	0.0	2.0	1.0	0	100	27.9	43.6	4
Max	5.0	17.0	19.0	17.0	16.0	75	100	100.0	100.0	94
Avg	3.1	7.0	7.4	7.0	6.0	13.9	100	72.2	81.2	30
Std. Dev.	1.3	4.6	6.2	4.6	4.6	20.7	0	25.9	19.5	30

As can be seen in the KDM generation time column, the time does not exceed 100 milliseconds in any of the cases, obtaining an average of 30 milliseconds and a standard deviation of 30. The results obtained show that the number of elements does not necessarily imply an increase in time. For example, the teleport algorithm ('teleport'), is one of the algorithms that generates more elements and relations, took 13 milliseconds, while the magic state distillation algorithm ('msd') took 88 milliseconds presenting a smaller number of elements.

5 CONCLUSIONS

This paper presents a reverse engineering technique that allows the representation of QASM programs in the KDM standard. The representation of the programs is performed by means of high-abstraction level models in a technology-agnostic way. The KDM models generated from the QASM programs available in the official Qiskit repository have been used to conduct a preliminary evaluation that shows suitability of this extension and contributes to its applicability in industry.

We can conclude that the major achievement of this proposal is to demonstrate that, despite the large number of quantum programming frameworks and languages, it is possible to agnostically model the information of quantum algorithms to be used in the software modernization of hybrid information systems. Because our task is to bring software modernization closer to the quantum paradigm, we represent the information extracted from the algorithms according to the KDM standard. KDM was not specifically developed for modelling quantum software, but its extension mechanism allows us to do so.

This proposal is framed in long-term research whose main objective is to adapt the process of software modernization for the combination of both classical and quantum software towards hybrid information systems.

Our future work has two separate but related paths. The first is the improvement of this technique, as possible improvements have been identified, including the implementation of the representation of oracles and other control structures in QASM. The second path concerns the implementation of the other phases of quantum software modernization, such as restructuring and forward engineering.

ACKNOWLEDGEMENTS

This work is part of the projects SMOQUIN (PID2019-104791RBI00) and QU-ASAP (PDC2022-133051-I00) funded by the Spanish Ministry of Science and Innovation (MICINN) and QHealth project (EXP 00135977/MIG-20201059), 2020 CDTI Missions Program (Center for the Development of Industrial Technology of the Ministry of Science and Innovation of Spain). We would like to thank all the aQuantum members, and particularly Guido Peterssen and, Pepe Hevia, for their help and support. Competitive Research Programme (CRP Award No. NRF-CRP 10-2012-03).

REFERENCES

- Andrew W. Cross, L. S. B., John A. Smolin, Jay M. Gambetta. (2020). OpenQASM3 ANTLRv4's Grammar specification. Retrieved from <https://qiskit.github.io/openqasm/grammar/index.html>
- Azeem Akbar, M., Rafi, S., & Khan, A. A. J. a. e.-p. (2022). Classical to Quantum Software Migration Journey Begins: A Conceptual Readiness Model. arXiv: 2209.05105.
- Bouwmeester, D., Pan, J.-W., Mattle, K., Eibl, M., Weinfurter, H., & Zeilinger, A. (1997). Experimental quantum teleportation. *Nature*, 390(6660), 575-579. doi:10.1038/37539
- De Stefano, M., Pecorelli, F., Di Nucci, D., Palomba, F., & De Lucia, A. (2022). Software engineering for quantum programming: How far are we? *Journal of Systems and Software*, 190, 111326. doi:<https://doi.org/10.1016/j.jss.2022.111326>
- Dunjko, V., & Briegel, H. J. J. R. o. P. i. P. (2018). Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *81(7)*, 074001.
- Durelli, R. S., Santibáñez, D. S. M., Marinho, B., Honda, R., Delamaro, M. E., Anquetil, N., & Camargo, V. V. d. (2014, 13-15 Aug. 2014). *A mapping study on architecture-driven modernization*. Paper presented at the Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014).
- Gamma, E., Helm, R., Johnson, R., Johnson, R. E., & Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*: Pearson Deutschland GmbH.
- Garhwal, S., Ghorani, M., & Ahmad, A. J. A. o. C. M. i. E. (2021). Quantum programming language: A systematic review of research topic and top cited languages. *28(2)*, 289-310.
- Gyongyosi, L., & Imre, S. (2019). A Survey on quantum computing technology. *Computer Science Review*, 31, 51-71. doi:<https://doi.org/10.1016/j.cosrev.2018.11.002>

- Houekpetodji, M. H., Anquetil, N., Ducasse, S., Djareddir, F., & Sudich, J. (2021). *Report From The Trenches A Case Study In Modernizing Software Development Practices*. Paper presented at the 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME).
- Imre, S., & Balazs, F. (2005). *Quantum Computing and Communications: an engineering approach*: John Wiley & Sons.
- ISO/IEC. (2009). Knowledge Discovery Meta-model (KDM). Retrieved from <https://www.iso.org/standard/32625.html>
- Jiménez-Navajas, L., Pérez-Castillo, R., & Piattini, M. (2020). Reverse Engineering of Quantum Programs Toward KDM Models. In *International Conference on the Quality of Information and Communications Technology* (pp. 249-262): Springer.
- Julian van Velzen, P. B., Sally Epstein, Michiel Boreel, Sam Genway, Preeti Yadav, Edmond Owen, Gireesh Kumar Neelakantaiah, Nadine van Son, Kary Bheemaiah, Prof Moez Draief, Jerome Buvat, Amol Khadikar, Gaurav Aggarwal. (2022). *Quantum technologies: How to prepare your organization for a quantum advantage now*. Retrieved from Capgemini Research Institute's Webpage: <https://www.capgemini.com/wp-content/uploads/2022/03/Final-Web-Version-Quantum-Technologies.pdf>
- Kazman, R., Woods, S. G., & Carrière, S. J. (1998). *Requirements for integrating software architecture and reengineering models: CORUM II*. Paper presented at the Proceedings fifth working conference on reverse engineering (Cat. No. 98TB100261).
- Li, H., Khomh, F., & Openja, M. (2021). *Understanding Quantum Software Engineering Challenges An Empirical Study on Stack Exchange Forums and GitHub Issues*. Paper presented at the 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME).
- McArdle, S., Endo, S., Aspuru-Guzik, A., Benjamin, S. C., & Yuan, X. (2020). Quantum computational chemistry. *Reviews of Modern Physics*, 92(1), 015003. doi:10.1103/RevModPhys.92.015003
- Nguyen, H. T., Usman, M., & Buyya, R. J. a. p. a. (2022). QFaaS: A Serverless Function-as-a-Service Framework for Quantum Computing.
- Paradauskas, B., Laurikaitis, A. J. I. t., & control. (2006). Business knowledge extraction from legacy information systems. 35(3).
- Parr, T. ANTLR's web page. Retrieved from <https://www.antlr.org/>
- Pérez-Castillo, R., De Guzman, I. G.-R., Piattini, M. J. C. S., & Interfaces. (2011). Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems. 33(6), 519-532.
- Pérez-Castillo, R., de Guzmán, I. G. R., & Piattini, M. (2011). Architecture-driven modernization. In *Modern Software Engineering Concepts and Practices: Advanced Approaches* (pp. 75-103): IGI Global.
- Pérez-Castillo, R., Jiménez-Navajas, L., & Piattini, M. (2021). QRev: migrating quantum code towards hybrid information systems. 1-30.
- Pérez-Castillo, R., Serrano, M. A., & Piattini, M. (2021). Software modernization to embrace quantum technology. *Advances in Engineering Software*, 151, 102933. doi:<https://doi.org/10.1016/j.advengsoft.2020.102933>
- Pérez-Delgado, C. A., & Perez-Gonzalez, H. G. (2020). *Towards a quantum software modeling language*. Paper presented at the Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops.
- Piattini, M., Peterssen, G., Pérez-Castillo, R., Hevia, J. L., Serrano, M. A., Hernández, G., . . . Murina, E. (2020). *The Talavera Manifesto for Quantum Software Engineering and Programming*. Paper presented at the QANSWER.
- Qiskit. (2022). OpenQASM3 Examples. Retrieved from <https://github.com/Qiskit/openqasm/tree/main/examples>
- Ulrich, W. (2010). Modernization Standards Roadmap. In *Information Systems Transformation* (pp. 45-64): Elsevier.
- Wang, R. (2021, NOV 30, 2021). Trends: Quantum Computing Market Cap Tops \$174 Billion. Retrieved from <https://www.constellationr.com/blog-news/trends-quantum-computing-market-cap-tops-174-billion>
- Weder, B., Barzen, J., Leymann, F., Salm, M., & Vietz, D. (2020). *The quantum software lifecycle*. Paper presented at the Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software.
- Wootters, W. K., & Zurek, W. H. J. P. T. (2009). The no-cloning theorem. 62(2), 76-77.