Software Quality Issues in Quantum Information Systems

Javier Verdugo [1], Moisés Rodríguez [1] and Mario Piattini [1]

[1] *Alarcos Research Group, University of Castilla-La Mancha, Pº de la Universidad, 4, Ciudad Real, 13071, Spain*

Abstract
Quantum computing is the technology of the 21st century. Quantum computers and environments are already offering great advantages when building advanced applications in finance, health, or logistics. However, if industry is to boost the large-scale production of quantum software, an adequate quality level must be achieved and assured. In this sense, it is very important to consider quantum software quality platforms and products, and to create an effective quality environment for quantum software. In this paper we will summarise some of these issues.

Keywords
Quantum Computing, Quantum Software Quality, Quantum Quality Environment.

1. Introduction

Quantum technology includes several research areas such as true random number generation, quantum information, atomic quantum clocks, quantum sensors, quantum simulators, quantum cryptography and security, quantum communication networks, and quantum internet. All these topics and most notably quantum computing, have attracted a lot of interest in recent years. There are countless interesting applications for these technologies, spanning several knowledge and business areas: [1] economics and financial services, chemistry, medicine and health, supply chain and logistics, energy, agriculture, etc. Quantum computing is also impacting different areas of computer science such as Cybersecurity [2] and Machine Learning and Artificial Intelligence [3]-[5]. The prospects for quantum computing are exciting, and extraordinary expectations are now driving a global effort to perfect quantum technologies [6]. It is believed that quantum computing could also bring a new "golden age" to software engineering [7].

The problem comes when we realise that in order to boost large-scale production of quantum software an adequate quality level is required [8], so that society can really benefit from the promising quantum applications that exist in the various domains. To achieve this, we should address all the areas of Software Engineering as defined in SWEBOK[2], i.e., Design, Software Construction, Software Maintenance, etc., and, specifically, Software Testing and Software Quality. Software Testing has been addressed by Miranskyy and Zhang [9], and Polo [10], while Software Governance has been addressed by Blanco and Piattini [11]. In this paper we focus on Software Quality; thus, Section 2 analyses the landscape of quantum software quality; Section 3 addresses some of the issues (models and metrics) to be considered when creating an efficient quality environment for quantum software; and, finally, Section 4 summarises the conclusions and future work.

Proceedings of the 2nd Quantum Software and Engineering Workshop (QSET'21), October 19, 2021, Virtual conference
EMAIL: javier.verdugo@uclm.es (A.1); moises.rodriguez@uclm.es (A.2); mario.piattini@uclm.es (A.3)
ORCID: 0000-0002-2526-2918 (A. 1); 0000-0003-2155-7409 (A. 2); 0000-0002-7212-8279 (A. 3)

[2] https://www.computer.org/education/bodies-of-knowledge/software-engineering

2. Quality Issues in Quantum Computing

In a quantum information system, there are several factors that influence the quality of the results: the quality of the quantum hardware, the quality of the quantum software (development and operational) platform, and the quality of the quantum software *per se*.

There are different types of quantum simulators and computers (adiabatic, gate-based, measurement-based, etc.); however, to date, most of them still present errors, hence their name: "NISQ—Noisy Intermediate-Scale Quantum" [12]. The *Committee on Technical Assessment of the Feasibility and Implications of Quantum Computing* has analysed the milestones in the evolution of quantum computers towards their current status as large, fault-tolerant modular quantum computers [13]. Moreover, the underlying technology —trapped ions, neutral-atom qubits, superconductors, quantum dots, semiconductor-based qubits, NV centre qubits, topological qubits, photons, etc.—has a decisive influence on the maturity level of quantum computers [14]. These different technologies present varying coherence times, gate latencies, gate fidelities, etc. [15].

Quantum hardware is not the only important matter for achieving "high quality" quantum information systems; software quality is also essential. In fact, Quantum Software Engineering (QSE) is an essential contribution to the success of quantum computing. One of the principles of the *Talavera Manifesto for Quantum Software Engineering and Programming* [16] establishes that "*QSE assures the quality of quantum software. Quality management for both, processes and products, is essential if quantum software with expected quality levels is to be produced*". Unfortunately, until now, quantum software quality issues have been largely disregarded.

Concerning quantum software processes, to date only a few interesting works have been proposed, mostly dealing with quantum life cycle processes [17]-[18], and the application of agile methodologies to quantum development [19]. In this paper we focus on the following quantum product software issues: quantum software development and execution platforms, and quantum software products.

2.1. Platform Quality

Sodhi & Kapur [20] have published an analysis of the impact of different quantum computing platforms on quality attributes and SDLC activities. They experimented with the main quantum programming platforms, examining how each one affects the main software quality characteristics: Availability, Interoperability, Maintainability, Manageability, Performance, Reliability, Scalability, Security, Testability, and Usability. Some of the characteristics dealt with in their study that impact the most on quality attributes, are:

1. A lower level of programming abstractions increasing code complexity, which has an impact on maintainability, testability, reliability, and availability.
2. Platform heterogeneity, which deteriorates software cohesion, affecting maintainability, reliability, robustness, reusability, and the manageability and testability of the system.
3. Remote software development and deployment, which make programming, testing, and debugging quantum programs slower, thus affecting maintainability and testability.
4. Dependency on the known quantum algorithms, thereby affecting the ability to perform enhancement and corrective maintenance, as well as testability and interoperability (with classical software).
5. Limited portability of software, which results in a lack of standardization in several areas, thus affecting availability, interoperability, maintainability, and scalability.
6. Lack of native quantum operating systems, thereby decreasing performance, manageability, reliability, scalability, and security.
7. Fundamentally different programming models, which can thus increase code complexity, so affecting maintainability, interoperability, security and testability.

This study could help us to evaluate the design decisions that need to be taken when constructing quantum software, and to carefully consider the influence of the given platform on every specific quality characteristic.

2.2. Software Quality

Most of the existing research efforts related to quality in quantum software have focused in general on quantum program verification [21] and specifically on verified compilation [22], verification protocols [23], relational verification of quantum programs [24], formal quantum programs description [25], formal verification and programs certification [26] and equivalence checking for quantum circuits [27].

In terms of metrics, most of the research deals with the measurement of the power of quantum computing - e.g., quantum volume [28]-[29], the TQF (Total Quantum Factor) and *wd* (width and depth) [30] etc. - but does not consider software features. One exception is Sicilia et al. [31], who carry out a preliminary study on the module structure and use of quantum gates in the libraries of Microsoft's quantum development platform QDK (Quantum Developer Kit) using Q#.

3. Towards a Quantum Software Quality Environment

A useful quantum software quality environment should contain, at least, a quantum quality model, a set of quantum software metrics, and a tool for supporting the automatic gathering and visualisation of the quantum metrics.

ISO/IEC 25010 [32] is the *de jure* quality standard for software products. In the ISO/IEC JTC1 SC7/WG6 Plenary Meeting held in June 2020, experts agreed that this standard should be adopted for Big Data and Artificial Intelligence, IoT, blockchain, cloud, Systems of Systems, XaaS, etc. To do so, a new "Quality Engineering Division" would be established. However, for the time being, none of these new models as yet cover quantum software.

We propose to introduce a specific quantum software quality model aligned with the general ISO/IEC standards family, considering the special nature of quantum computing. Firstly, the applicability to quantum software of the existing characteristics (performance efficiency, maintainability, portability, compatibility, etc.) must be considered. From our experience in using quantum programming languages with software practitioners and students, we have seen, for example, that understandability is a subcharacteristic which needs to be strongly modified for quantum programs. In fact, it can be difficult to understand not only this new computing paradigm, but also the specific quantum software design and programs.

In a second step, research must be undertaken on new quality characteristics and aspects of quantum software. For example, even if ISO/IEC 25010 does not consider either accuracy or precision as being software quality characteristics, decoherence (due to errors in quantum computers) means that these characteristics are in fact very important. At present, and for the foreseeable future, only Noisy Intermediate-Scale Quantum (NISQ) technology is going to be available. Hence, we must consider both the current quantum technologies and future ones (such as quantum fault tolerance computers) when defining all the quality model characteristics.

Although there is a lot of research on metrics for classical conceptual models, there is none for quantum software. Firstly, it is important to define metrics at a conceptual level (i.e., quantum technology-agnostic ones). For this purpose, the conceptual model could be a UML extension for describing quantum systems [33] or a quantum circuit model [34], which abstract quantum information as measurement entities.

Furthermore, metrics need to be defined at the logical (technology-specific) level, taking into account the different quantum programming languages that currently exist [35]. Both at a conceptual and a logical level, metrics must consider the overhead that arises due to the use of the different quantum characteristics (entanglement, superposition, non-cloning, etc.), and the different quantum gates (Hadamard, Pauli X, Y and Z, CNOT, Controlled-Z, Toffoli, etc.).

To support these metrics calculations, tools [36] should be created, extending (when possible) existing open-source tools or following a metamodel approach.

4. Conclusions and Future Work

Quantum computers have the potential to solve the type of tasks that today we do not even dare dream of and which classical computers will never be able to solve (EQF, 2020). In fact, quantum computing speeds up the process of solving algorithms that require massive parallel computations, and so allows us to better simulate nature.

However, to achieve all these benefits, quantum software has to be developed in an appropriate way. We must commence with quantum software engineering now, so as to be prepared for the future and also so as to avoid low quality quantum software with errors and productivity problems in the interim. For that end, we need a comprehensive quantum software quality environment, including quality models, metrics, and tools. This kind of environment could be useful not only in creating and developing quantum software but also in overall modernisation and reengineering efforts [37].

Acknowledgements

We would like to thank all the aQuantum members, especially Guido Peterssen and Pepe Hevia, for their help and support. This work was partially funded by the "QHealth: Quantum Pharmacogenomics Applied to Aging" project, the 2020 CDTI Missions Program (Center for the Development of Industrial Technology of the Ministry of Science and Innovation of Spain), and the GEMA project (Consejería de Educación, Cultura y Deportes de la Junta de Comunidades de Castilla La Mancha, y Fondo Europeo de Desarrollo Regional FEDER) under Grant SBPLY/17/180501/000293.

References

[1] IDB. Quantum technologies. Digital transformation, social impact, and cross-sector disruption. Interamerican Development Bank. (2019)

[2] P. Wallden and E. Kashefi. CyberSecurity in the Quantum Era. Communications of the ACM 62 (4), April. (2019) 120-129.

[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe and S. Lloyd. Quantum machine learning, Nature 549, 195. (2017)

[4] S. Garg and G. Ramakrishnan. Advances in Quantum Deep Learning: An Overview. (2020) arXiv:2005.04316v1.

[5] Y. Zhang and Q. Ni. Recent advances in quantum machine learning. Quantum Engineering, 2: e34. (2020) https://doi.org/10.1002/que2.34.

[6] T.S. Humble and E.P. DeBenedictis. Quantum Realism. IEEE Computer 52 (6). (2019) 13-17.

[7] M. Piattini, G. Peterssen and R. Pérez-Castillo. Quantum Computing: A New Software Engineering Golden Age. ACM SIGSOFT Software Engineering Newsletter 45 (3), June. (2020) 12-14.

[8] M. Piattini, M. Serrano, R. Pérez-Castillo, G. Peterssen and J.L. Hevia. Towards a Quantum Software Engineering. IT Professional, vol. 23, no. 1 Jan.-Feb. (2021) 62-66. doi: 10.1109/MITP.2020.3019522.

[9] A. Miranskyy and L. Zhang. On Testing Quantum Programs. Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results. (2019) 57-60. doi:/10.1109/ICSE-NIER.2019.00023. arXiv:1812.09261v1

[10] M. Polo. Quantum Software Testing. QANSWER 2020 QuANtum SoftWare Engineering & pRogramming. Proceedings of the 1st International Workshop on the QuANtum SoftWare Engineering & pRogramming, Talavera de la Reina, Spain, February 11-12. (2020) 57-63.

[11] M. Á. Blanco and M. Piattini. Adapting COBIT for Quantum Computing Governance. In M. Shepperd, F. Brito e Abreu, A. Rodrigues da Silva, & R. Pérez-Castillo (Eds.), Quality of Information and Communications Technology. Springer International Publishing. (2020) 274–283. https://doi.org/10.1007/978-3-030-58793-2_22

[12] J. Preskill. Quantum Computing in the NISQ era and beyond. Quantum 2, 79. (2018) doi.org/10.22331/q-2018-08-06-79.

[13] E. Grumbling and M. Horowitz. Quantum Computing Progress and Prospects. Washington DC, The National Academies Press. (2019)
[14] EQF. Strategic Research Agenda. European Quantum Flagship. February (2020).
[15] S. Resch and U.R. Karpuzcu. Quantum Computing: An Overview Across the System Stack. (2019) arXiv:1905.07240v1.
[16] M. Piattini, G. Peterssen, R. Pérez-Castillo, J.L. Hevia et al. The Talavera Manifesto for Quantum Software Engineering and Programming. QANSWER 2020 QuANtum SoftWare Engineering & pRogramming. Proceedings of the 1st International Workshop on the QuANtum SoftWare Engineering & pRogramming, Talavera de la Reina, Spain, February 11-12, (2020). http://ceur-ws.org/Vol-2561/paper0.pdf.
[17] N. Dey, M. Ghosh, S. Samir and A. Chakrabarti. The Quantum Development Life Cycle arXiv:2010.08053v1 cs.ET 15 Oct (2020).
[18] B. Weder, J. Barzen, F. Leymann, M.O. Salm, D. Vietz. The Quantum software lifecycle. APEQS 2020: Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software November (2020) 2–9. https://doi.org/10.1145/3412451.3428497.
[19] G. Hernández and C.A. Paradela. Quantum Agile Development Framework, QUATIC 2020. (2020) 284-291
[20] B. Sodhi and R. Kapur. Quantum Computing Platforms: Assessing Impact on Quality Attributes and SDLC Activities (Accepted in ICSA 2021) January (2021), DOI: 10.13140/RG.2.2.20190.66886/1.
[21] R. Rand. Research Statement: Languages, Verification and Compilation for the Quantum Era. (2020) https://pdfs.semanticscholar.org/303b/7bec1053f132c7250dc50d9257da3aa8f39a.pdf?_ga=2.137759792.102012242.1590658429-1729744265.1580375978
[22] R. Rand, J. Paykin and S. Zdancewic. QWIRE Practice: Formal Verification of Quantum Circuits in Coq.14th International Conference on Quantum Physics and Logic (QPL). Bob Coecke and Aleks Kissinger (Eds.): EPTCS 266, (2018) 119–132.
[23] A. Gheorghiu, T. Kapourniotis and E. Kashefi. Verification of quantum computation: An overview of existing approaches. Theory of Computing systems. (2018) arXiv:1709.06984v2.
[24] G. Barthe, J. Hsu, M. Ying, N. Yu and L. Zhou. Relational Proofs for Quantum Programs. Proc. ACM Program. Lang., 4, No. POPL, Article 21. (2020)
[25] C. Cartiere. Formal Quantum Software Engineering. Introducing the Formal Methods of Software Engineering to Quantum Computing. (2020) https://www.researchgate.net/publication/338631774_Formal_Quantum_Software_Engineering_Introducing_the_Formal_Methods_of_Software_Engineering_to_Quantum_Computing.
[26] C. Chareton, S. Bardin, F. Bobot, V. Perrelle and B. Valiron. Toward certified quantum programming. (2020) arXiv:2003.05841v1.
[27] L. Burgholzer and R. Wille. Advanced Equivalence Checking for Quantum Circuits. (2020) arXiv:2004.08420v1.
[28] L. S. Bishop, S. Bravyi, A. Cross, J.M. Gambetta and J. Smolin. Quantum Volume. March 4, (2017). https://www.semanticscholar.org/paper/Quantum-Volume-Bishop-Bravyi/650c3fa2a231cd77cf3d882e1659 ee14175c01d5
[29] A.W. Cross, L.S. Bishop, S. Sheldon, P.D. Nation and J.M. Gambetta. Validating quantum computers using randomized model circuits. Phys. Rev. A 100, 032328 – Published 20 September (2019), doi:10.1103/PhysRevA.100.032328.
[30] M. Salm, J. Barzen, F. Leymann and B. Weder. About a Criterion of Successfully Executing a Circuit in the NISQ Era: What $wd \ll 1/\epsilon$eff Really Means. Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software (APEQS '20), November 13, (2020). Virtual, USA. ACM, New York, NY.
[31] M.A. Sicilia, S. Sánchez, M. Mora and E. García. On the Source Code Structure of Quantum Code: Insights from Q# and QDK. QUATIC 2020, (2020) 292-299.
[32] ISO. ISO/IEC 25010 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models. Ginebra, International Organization for Standarization. (2010)

[33] C.A. Pérez-Delgado and H.G. Pérez-González. Towards a Quantum Software Modeling Language. ICSE (Workshops) (2020), 442-444.
[34] E. Rieffel and W. Polak. Quantum Computing. A Gentle Introduction. Cambridge, Massachusetts, The MIT Press. (2011)
[35] S. Garhwal, M. Ghorani and A. Ahmad. Quantum Programming Language: A Systematic Review of Research Topic and Top Cited Languages. Archives of Computational Methods in Engineering. (2019) https://doi.org/10.1007/s11831-019-09372-6
[36] M. Rodríguez, M. Piattini and C. Ebert. Software Verification and Validation Technologies and Tools. IEEE Software 36(2). (2019) 13-24.
[37] R. Pérez-Castillo, M.A. Serrano and M. Piattini. Software modernization to embrace quantum technology. Adv. Eng. Softw. 151: 102933. (2021).