1-2-3 Reproducibility
for Quantum Software Experiments

Wolfgang Mauerer
Technical University of
Applied Sciences Regensburg
Siemens AG, Corporate Research
wolfgang.mauerer@othr.de

Stefanie Scherzinger
Chair of Scalable Database Systems
University of Passau
Passau, Germany
stefanie.scherzinger@uni-passau.de

Abstract—**Various fields of science face a reproducibility crisis. For quantum software engineering as an emerging field, it is therefore imminent to focus on proper reproducibility engineering from the start. Yet the provision of reproduction packages is almost universally lacking. Actionable advice on how to build such packages is rare, particularly unfortunate in a field with many contributions from researchers with backgrounds outside computer science. In this article, we argue how to rectify this deficiency by proposing a 1-2-3 approach to reproducibility engineering for quantum software experiments: Using a meta-generation mechanism, we generate DOI-safe, long-term functioning and dependency-free reproduction packages. They are designed to satisfy the requirements of professional and learned societies solely on the basis of project-specific research artefacts (source code, measurement and configuration data), and require little temporal investment by researchers. Our scheme ascertains long-term traceability even when the quantum processor itself is no longer accessible. By drastically lowering the technical bar, we foster the proliferation of reproduction packages in quantum software experiments and ease the inclusion of non-CS researchers entering the field.**

Index Terms—**Reproducibility engineering, quantum software engineering**

I. Introduction

The ACM reproducibility guidelines (v1.1) [1] consider an experiment *reproducible* when a different team with a different experimental setup is able to confirm the published results.[1] Despite the universally acknowledged importance of reproducibility, the vast majority of researchers agree that we are facing a reproducibility crisis [2], [3] in almost all domains of science, including quantum computing research [4], [5], [6], [7].

In the emerging field of quantum software (QSW) engineering, reproducibility engineering has also been recognised as a grand challenge [4], [5], [6], [7], with first contributions on managing the reproducibility of software bugs [6], [4], or on exploring the parameter search space for quantum optimisation tasks [8]. Yet so far, little actionable advice has been given on how to engineer QSW experiments for reproducibility. As we will argue shortly, despite the general credo that reproducibility

[1]The ACM has repeatedly changed the definition of their reproducibility terminology in an incompatible way; we are aware of the arising confusion.

is important, at this moment in time, published articles that are accompanied by reproduction packages are rare.

Building a working *reproduction package* goes beyond providing a DOI to some repository hosting data, code, and setup instructions. Rather, a gold-standard reproduction package [9] bundles all research artefacts required to conduct the experiment (such as source code, libraries, or input data), and contains a dispatcher script that allows for executing and evaluating the experiment via a single command.

Quantum-specific challenges: Building reproduction packages for QSW shares many challenges of classic software engineering, most importantly managing complex software stacks that non-trivially interact with hardware.

Reproducibility challenges specific to QSW experiments are plentiful [10]: Quantum computing hardware is usually provided as a cloud service, with vendor-controlled access and configurations. Reproducers may not have ready access to the very same machine (especially after prolonged periods of time). Even if they do, they are very likely to find the configuration changed, given that most quantum hardware is still in its infancy and subject to change even during the operational phase.

Apart from a unique model/type specification of the employed machine, it is necessary to provide information on (a) input generation methods, (b) qbit counts, (c) connectivity topology, (d) any methods used to transform inputs, (e) approaches used to map/embed logical qbits onto physical qbits, (f) postprocessing methods and utilities employed, and (g) programming/initialisation and readout times. Quantum software experiments may involve manual tuning, which necessitates specifying any employed heuristics/policies. Finally, details on how runtimes are measured are important, especially in cloud settings that may involve interference by access schedulers.

Synopsis: We quantitatively review the state of reproducibility in QSW experiments, and recognise a need for action. We then introduce a customisable *meta package template* that generates end-to-end, one-click reproductions with adaptation examples for common quantum hardware. It can be found at the accompanying website/DOI), together with a video tutorial. Only little input is required from the scientists.

Vision: The ease-of-use of our reproduction package template enables the proliferation of working reproduction packages for QSW experiments. Moreover, lowering the thresh-

old for building reproduction packages can foster skills in QSW engineering, an area difficult to master for newcomers [11].

Venue	Year	# Papers	# Exp	# Src	# Repro
QSA@ICSA	2021	4	2	1	0
Q-SET@QW	2020	6	2	0	0
Q-SET	2021	4	2	1	0
QCS@SAC	2021	11	10	8	0
APEQS@FSE	2020	4	1	0	0
QTOP@Netsys	2019	18	12	2	1
Q-SE@ICSE	2020	8	3	2	0
Q-SE@ICSE	2021	8	3	1	1

II. STATE OF THE ART

State of reproducibility. Table I provides an overview about the state of reproducibility in quantum software. Based on eight workshops from 2019–2021, where we currently observe most activities related to QSW, we reviewed 63 papers (column "# Papers"; detailed results are on the accompanying website). We classified them according to whether they describe artefacts (column "# Exp"; algorithmic source code or experiments to be run on quantum hardware or simulators) that should be provided in a reproducible form. We also state the number of papers that *at least* provide source code on a non-permanent software forge like GitHub (column "# Src"), and the fraction of papers that provide a reproduction package (column "# Repro"; we count anything provided under a DOI-safe permanent location, even if it does not meet the standards suggested in Ref. [1]).

The conclusion is simple: Although most published research is concerned with executing code and simulations, only slightly more than half of the publications provide source code *at all*, and only *two* papers come with artefacts on a DOI-safe location! Clearly, there is a mismatch between desirable reproducibility qualities as mandated by professional associations (and, therefore, the community itself), and scientific day-to-day reality.
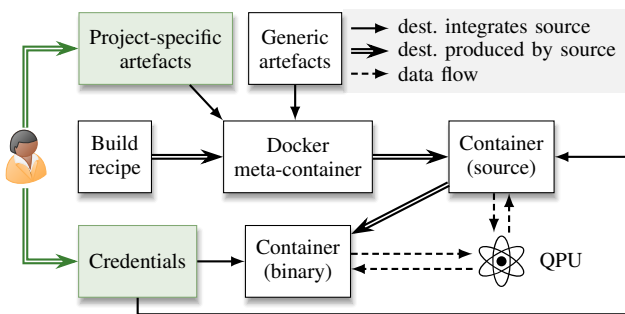

Fig. 1. Components and workflow for 1-2-3 reproducibility engineering: The scientist provides project-specific artefacts (green). A docker meta-container then generates a reproduction package consisting of a source container and a pre-built binary docker container. Both containers can trigger the QSW experiment, record measurement and configuration data, and analyse results. Credentials for accessing commercial quantum hardware and cloud services are not encoded into the reproduction package, but specified as parameter.

III. A TEMPLATE PROPOSAL

To make QSW experiments reproducible with little extra effort for researchers, we suggest a generic, adaptable template (with full source available on the accompanying website) as reference for QSW experiments as illustrated in Fig. 1. It includes examples for dealing with DWave and IBMQ systems. Based on the supplied artefacts, the generated package comprises (a) the complete source code for the calculations performed in the paper, (b) results obtained on quantum computers (to ensure traceability beyond quantum processor availability) and via classical simulation, (c) a concise documentation of each reproduction step, (d) a "one-click" dispatcher to run the pipeline, (e) ideally, means of generating the underlying research paper, including graphs and tables.

The replication package in both, source and pre-built binary form, is intended to be made available at a DOI-safe, long-term stable location. The former is easier to extend by peer researchers, but assumes external components on volatile locations are still available. The latter ensures long-term reproducibility, since it does not have external dependencies.

To avoid impediments for adoption that might arise from technologies like docker—many researchers in QSW do not have computer science backgrounds, and even if, it might not be systems-centric—, we suggest this generative approach.

IV. CONCLUSION

Reproducibility of experiments is key in scientific research. Yet the adoption of methods to ensure this quality is currently sub-par in the quantum software literature. Since the field is still in relative infancy, we propose sustainable rectification of the situation from the start, by providing a generic scheme that generates reproduction packages that are tailored to quantum software experiments, and that can be used with minimal effort.

REFERENCES

[1] Association for Computing Machinery. (2020, 08) Artifact review and badging (version 1.1). [Online]. Available: https://www.acm.org/publications/policies/artifact-review-and-badging-current

[2] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature News*, vol. 533, no. 7604, p. 452, May 2016.

[3] B. R. Jasny, G. Chin, L. Chong, and S. Vignieri, "Again, and Again, and Again …," *Science*, vol. 334, no. 6060, pp. 1225–1225, 2011.

[4] J. Campos and A. Souto, "QBugs: A Collection of Reproducible Bugs in Quantum Algorithms and a Supporting Infrastructure to Enable Controlled Quantum Software Testing and Debugging Experiments," in *Proc. Q-SE@ICSE 2021*, 2021, pp. 28–32.

[5] J. Zhao, "Quantum Software Engineering: Landscapes and Horizons," *CoRR*, vol. abs/2007.07047, 2020.

[6] P. Zhao, J. Zhao, Z. Miao, and S. Lan, "Bugs4Q: A Benchmark of Real Bugs for Quantum Programs," *CoRR*, vol. abs/2108.09744, 2021.

[7] M. Fingerhuth, T. Babej, and P. Wittek, "Open source software in quantum computing," *PLOS ONE*, vol. 13, no. 12, p. e0208561, Dec. 2018.

[8] R. Shaydulin, K. Marwaha, J. Wurtz, and P. C. Lotshaw, "QAOAKit: A Toolkit for Reproducible Study, Application, and Verification of the QAOA," *CoRR*, vol. abs/2110.05555, 2021.

[9] B. J. Heil, M. M. Hoffman, F. Markowetz, S.-I. Lee, C. S. Greene, and S. C. Hicks, "Reproducibility standards for machine learning in the life sciences," *Nature Methods*, vol. 18, no. 10, pp. 1132–1135, Aug. 2021.

[10] C. C. McGeoch, "Principles and guidelines for quantum performance analysis," in *QTOP@NetSys*, ser. Lecture Notes in Computer Science, vol. 11413. Springer, 2017, pp. 36–48.

[11] R. Shaydulin, C. Thomas, and P. Rodeghero, "Making Quantum Computing Open: Lessons from Open Source Projects," in *Proc. International Conference on Software Engineering Workshops*, 2020, p. 451–455.