Software engineering for 'quantum advantage'

Luis S. Barbosa
lsb@di.uminho.pt
INL - International Iberian Nanotechnology Laboratory
UNU-EGOV - United Nations University
Universidade do Minho
Braga, Portugal

ABSTRACT

Software is a critical factor in the reliability of computer systems. While the development of hardware is assisted by mature science and engineering disciplines, software science is still in its infancy. This situation is likely to worsen in the future with quantum computer systems. Actually, if quantum computing is quickly coming of age, with potential groundbreaking impacts on many different fields, such benefits come at a price: quantum programming is hard and finding new quantum algorithms is far from straightforward. Thus, the need for suitable formal techniques in quantum software development is even bigger than in classical computation. A lack of reliable approaches to quantum computer programming will put at risk the expected quantum advantage of the new hardware. This position paper argues for the need for a proper quantum software engineering discipline benefiting from precise foundations and calculi, capable of supporting algorithm development and analysis.

KEYWORDS

quantum computing, formal methods, software engineering

ACM Reference Format:
Luis S. Barbosa. 2020. Software engineering for 'quantum advantage'. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3387940.3392184

1 THE CLAIM

Arguably quantum computing is coming of age. With the race for quantum rising between major IT players, and the announcement of new prototype, proof-of-concept machines up to 50 qubits, it seems we are in the verge of a real shift. For the first time, the viability of quantum computing may be demonstrated in a number of real problems extremely difficult to handle, if possible at all, classically, and its utility discussed across industries. In a sense, Feynman's dream of letting Nature, suitably engineered, compute for us through its own natural quantum behaviour, seems to be closer, even if the project of a universal quantum computer has still a long way to go. In the somehow emphatic language of the media, a 'second quantum revolution' is quickly approaching. It is characterised by the ability to harness the most weird quantum phenomena, namely *superposition* and *entanglement*, as computational resources, with practical advantage. In this move the role of software, and its engineering, cannot be underestimated.

Software is indeed a critical factor in the reliability of computer systems, a situation which is likely to worsen as quantum systems will be more difficult to program and test. Lack of trustworthy approaches to software development may put at risk the expected 'quantum advantage? of the new hardware, to use a term popularised by industry. This entails the need for *reworking and extending the whole of classical software engineering into the quantum domain so that programmers can manipulate quantum programs with the same ease and confidence that they manipulate today's classical programs*, as suggested as a research 'grand challenges' in the initial years of the millennium [14].

Software engineering is an all-encompassing domain, ranging from requirements and architectures, to design, development, verification and deployment. Even narrowing its scope to focus exclusively on the design of algorithms, such re-working is pressing. The set of primitive techniques in quantum algorithmics increased over the past decade, exploring quantum effects in a number of unsuspected ways. But still quantum programming is hard, finding new and effective quantum algorithms is far from straightforward, some useful metaphors may still lack.

Moreover, most current quantum algorithms assume an ideal quantum computer with many qubits that can hold information indefinitely. We are not yet there. In the short term, the challenge is to find real-world problems and applications that can benefit from the small, 'noisy' quantum computers that will soon be available.

Irrespective of one's own assessment of what the future might bring for this area, we believe it is time to discuss an agenda for a solid, rigorous software engineering discipline for quantum systems. Similarly to what happened in classical computation, such a discipline will greatly benefit from mature formal techniques, in the confluence of several mathematical domains (logic, algebra, topology, probability, category theory), able to conceptualise and predict behaviour of quantum computational systems, and to provide a rich, formal framework for their specification, analysis and development.

The conceptualisation of quantum computing predated its technological realisation: in a way physicists are making it happen. Similarly, in the 1930's, Turing machines anticipated digital computers. It seems history is repeating itself. Differently, however, from what happened before, we have now the chance to get theory

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3392184

in place before technologies emerge and popularise. The remaining of this paper briefly discusses a number of topics that may be relevant for this agenda.

2 THE RESEARCH CHALLENGES

Progress in quantum computing, related algorithmic techniques and applications, cannot ignore the fact that current methods and tools for quantum software development are still highly fragmentary and fundamentally 'low-level'. Reasoning directly with quantum gates is as limited as assembling logical gates in classical algorithm design. It sweeps under the carpet all key ingredients of a mature software engineering discipline: compositionality, abstraction, refinement, high-order and property-enforcing type schemes.

On the other hand, the standard mathematical formulation of quantum mechanics in terms of Hilbert spaces, and the associated von Neumann approach to its logical structure, is unable to provide a sufficiently abstract framework for specifying and analysing quantum processes and, in particular, to incorporate classical, macroscopic noise into the picture, in an effective, not implicit way.

In this context, we identify the following main issues around which any roadmap for a Software Engineering discipline meeting rigorous scientific standards should be structured: i) how (quantum) systems are *modelled*, ii) how models are *composed*, and finally, ii) how *properties* of their behaviours are anticipated, expressed and verified. More concretely, research on an Engineering discipline relevant for quantum computing should seek for

(1) appropriate *semantic structures*, able to comply with different types of classical control (non deterministic, probabilistic, continuous) and quantum data, as well as to capture a suitable notion of program approximation upon which a theory of quantum program refinement and equivalence can be based;
(2) an *algorithmic calculus*, stemming from the semantics above, for the systematic derivation of quantum programs in a compositional way;
(3) a new family of *dynamic logics*, parametric in the interfering (classical) computational paradigm, to support the formulation of contracts for quantum algorithms and their compositional verification;
(4) a *framework for coordination* of distributed quantum computational systems — a main requirement for obtaining optimally responsive global quantum networks, but currently largely overlooked.

The next section details specific research paths/strategies to realize these objectives. In this exercise, one must taken into account that quantum computing is significantly different from its classical counterpart. Even more, its evolution has influenced the way the proper foundations of quantum mechanics are currently perceived. The phenomenon of entanglement is a much-cited example: a paradox in the 1930's, a theorem thirty years later, with Bell's non-locality result, and, finally, three decades afterwards, a main computational resource to explain e.g. teleportation protocols. On the other hand, the patrimony collected over time in classical software engineering is most valuable at two levels. First to set a collection of themes to look at in the new setting: compositionality, abstraction, etc. Secondly, and most concretely, because it seems likely that, at least for a reasonable time span, classical and quantum software will coexist and the engineering process will need to deal with their hybrid development seamlessly.

3 RESEARCH DIRECTIONS

Systems' *models*, *architectures*, and *properties* constitute three main topics unavoidable in any roadmap to Software Engineering. In broad terms they capture three substantive perspectives which underly any effective design discipline: abstraction, compositionality and the ability to reason formally on the engineered artefacts. From our perspective, they are equally relevant to a research agenda for quantum software engineering. This section proposes a number of concrete research directions in each of these three areas, regarded from a formal methods point of view. We do not claim their centrality: at this stage what seems important is to discuss new ideas and illustrate what can be done.

3.1 Models

Models are pervasive in the engineering practice, and the software domain is not an exception. Irrespective of the myriad of (textual, diagrammatic, formal, etc.) notations used in practice, models should always be understood in the sense they are in *e.g.* school physics problem-solving. There, once a problem is understood, a mathematical model is built as an appropriate abstraction, on top of which one reasons about the behaviour of the system until a 'solution' is found.

Different effects in computational models, for example partiality, non determinism, probabilism, etc, are formally captured by monads, as in e.g. [4]. A number of proposals for monads capturing quantum features have recently appeared [1, 8, 10] which may lay the (semantic) foundations for quantum programming languages in the spirit of Moggis's pioneering work [11].

In particular, this may be a way to bring quantum and cyber-physical programming together. Actually, recent developments on the semantics of (classical) hybrid components introduced a new monad capturing continuous evolution in a topological setting which caters for stability aspects [12]. This can be lifted to the quantum domain seeking for a coherent, unified view of both classical and quantum hybrid devices and their interaction. The relevance of this comes form the fact that quantum devices will soon become part of major cyber-physical systems, entailing the need for precise behavioural specifications of their interaction with (macroscopic) continuous processes. Indeed, harnessing superposition states, easily affected by context, to produce very precise sensors, or using solid-state quantum sensors to measure very small magnetic fields, are technological possibilities identified in the 2016 Manifesto [13] as short term developments.

3.2 Architectures

Software architecture emerged as a proper discipline within Software Engineering from the need to explicitly consider, in the development of increasingly larger and more complex systems, their overall structure, organisation, and emergent behaviour. As a model, an architecture acts as an abstraction of a system that suppresses details of its constituents, except for those which affect the ways they use, are used by, relate to, or interact with other components.

Architectural approaches based on exogenous coordination models, as popularised in *e.g.* the Reo framework [2], seem worth to explore, given its enforcement of a clear separation of computation and interaction *loci*, and the intrinsic, *sui generis* parallelism inherent to the quantum computational model. Actually, the combination of quantum computational systems is largely overlooked despite being a fundamental ingredient of future architectures, in which distributed quantum computations may run, possibly aided by long-range classical communication, over classical or quantum data. A first challenge in this direction will aims at exploring what an algebra of mixed classical and quantum systems could be. Note that, as intermediate states in a quantum computation cannot be observed, even sequential composition is quite different from the classical case. Entanglement, on its turn, requires a new understanding of parallel composition. On the one hand, it restricts the interleaving abstraction, widely used in the analysis of classical concurrent processes. On the other, it may bring to scene new synchronisation mechanisms.

In a sense the mathematical structure underlying 'categorical quantum mechanics' [9] already possesses the basic ingredients for a calculus of quantum processes: composition (via tensor), measurements of entangled states (allowed by the compact closed structure), feedback (through dagger) and probabilistic branching (via biproducts), and last but not least, a formal diagrammatic notation [5].

3.3 Properties.

A plethora of logics is used in Software Engineering to support the specification of systems' requirements and properties, as well as to verify whether, or to what extent, they are enforced in specific implementations. Broadly speaking, the logics of dynamical systems are *modal*, *i.e.* they provide operators which qualify formulas as holding in a certain *mode*. In mediaeval Scholastics such modes represented the strength of assertion (e.g. 'necessity' or 'possibility'). In temporal reasoning they can refer to a future or past instant, or a collection thereof. Similarly, one may express epistemic states (e.g. 'as everyone knows'), deontic obligations (e.g. 'when legally entitled'), or spatial states (e.g. 'in every point of a surface'). Dynamic logic [7] is a well-known, successful modal logic to reason about (classical) programs and establish software correctness. Its extension to the quantum domain is not new (see references below), but much is required to tune its application to large, complex (quantum) systems. For example, how can contract-based design, a so successful paradigm in the classic software engineering practice, be extended to the quantum domain?

The work of Baltag and Smets [3], Panagaden [6] and Ying [15], proposes different dynamic logics able to express and reason about quantum programs. Some of our recent work is focused on a very general framework for the development of dynamic logics 'on-demand', i.e. parametric on program constructs and the structure of truth spaces. Theoretical aspects of such logics, such as complexity, decidability and calculi, also come up parametrically. Both quantum and classical (probabilistic) features can be combined in this framework, from which a corresponding family of Hoare logics for the quantum domain can be generated. In particular, three research challenges emerge: i) the characterisation of a contract-based development discipline for assertional reasoning about quantum programs; ii) the proposal of a dynamic logic incorporating (classical) noise at the expression level to reason about probabilistically controlled fault-tolerance in quantum programs; and, finally, iii) the identification of a semantic bridge between 'categorical quantum mechanics', which may be seen as a type theoretic form of quantum logic, and dynamic logics supporting assertional reasoning.

4 CONCLUDING

Although information technology became ubiquitous in modern life long before a solid scientific methodology, let alone formal foundations, has been put forward, the ultimate goal of a software engineering discipline is the development of methods, techniques and tools for formal – and preferably automatic – analysis and verification of computational systems.

Our starting point is that, just like classical computation, quantum algorithmics will greatly benefit from a mathematically based approach, able to conceptualise, and predict behaviour, and to provide a rich, formal framework for specifying, developing and verifying quantum algorithms. Such foundations are produced in the confluence of several mathematical disciplines and lessons learnt from the Software engineering practice. A few challenges and possible research direction were proposed in this paper to foster what we see as an urgent discussion.

Acknowledgments. This work was supported by ERDF, through COMPETE 2020 Programme, and FCT (Fundação para a Ciência e a Tecnologia), the Portuguese funding agency, within project POCI-01-0145-FEDER-030947.

REFERENCES

[1] S. Abramsky, R. Barbosa, N. Silva, and Z. Zapata. 2017. The Quantum Monad on Relational Structures. In *42 Symp. Mathematical Foundations of Computer Science (LIPIcs)*, K. G. Larsen *et al* (Ed.), Vol. 83. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 35:1–35:19.
[2] F. Arbab. 2004. Reo: a channel–based coordination model for component composition. *Mathematical Structures in Comp. Sci.* 14, 3 (2004), 329–366.
[3] A. Baltag and S. Smets. 2011. Quantum logic as a dynamic logic. *Synthese* 179, 2 (2011), 285–306.
[4] L. S. Barbosa. 2003. Towards a calculus of state-based software components. *"Jour. Universal Comp. Sci."* 9, 8 (2003), 891–909.
[5] B. Coecke and A. Kissinger. 2017. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning.* Cambridge University Press.
[6] E. D'Hondt and P. Panangaden. 2006. Quantum weakest preconditions. *Mathematical Structures in Computer Science* 16, 3 (2006), 429–451.
[7] David Harel, Dexter Kozen, and Jerzy Tiuryn. 2000. *Dynamic Logic.* MIT Press.
[8] I. Hasuo and N. Hoshino. 2017. Semantics of higher-order quantum computation via geometry of interaction. *Ann. Pure Appl. Logic* 168, 2 (2017), 404–469.
[9] C. Heunen and J. Vicary. 2019. *Categories for Quantum Theory.* Oxford University Press.
[10] B. Jacobs, J. Mandemaker, and R. Furber. 2016. The expectation monad in quantum foundations. *Inf. Comput.* 250 (2016), 87–114.
[11] E. Moggi. 1991. Notions of Computation and Monads. *Information and Computation* 93, 1 (1991), 55–92.
[12] R. Neves, L. S. Barbosa, D. Hofmann, and M. A. Martins. 2016. Continuity as a computational effect. *J. Log. Algebr. Meth. Program.* 85, 5 (2016), 1057–1085. https://doi.org/10.1016/j.jlamp.2016.05.005
[13] QUROPE – Quantum Information Processing and Communication in Europe. 2016. *Quantum Manifesto: A new era of technology.* Available from `qurope.eu`.
[14] S. Stepney, S. Abramsky, A. Adamatzky, C. G. Johnson, and J. Timmis. 2008. Grand Challenge 7: Journeys in Non-Classical Computation. In *Visions of Computer Science*, E. *et al* Gelenbe (Ed.). BCS, 407–421.
[15] M. Ying, S. Ying, and X. Wu. 2017. Invariants of quantum programs: characterisations and generation. In *44th ACM SIGPLAN Symp. Principles of Programming Languages, POPL, 2017*, G. Castagna and A. D. Gordon (Eds.). ACM, 818–832.