Quantum theory, SDLC, testing tools and use case example

Source Book:

Quantum Software: Aspects of Theory and System Design, Iaakov Exman et al,ISBN 978-3-031-64135-0 ISBN 978-3-031-64136-7 (eBook) – Open Access , https://doi.org/10.1007/978-3-031-64136-7

Need:

There is a need to look into the SDLC that is suitable for a wide range of targets and platforms. This book offers insights on this and the prevailing limitations of the quantum circuit and alternate software design approach that uses more abstraction and Meta models to enhance the portability that is required for the QAI Hub that we are working on.

Tools:

QMutPy, QChecker,QuCAT ,QuSBT,QUITO, Bugs4Q, QDiff, LintQ, QSmell , QuMU

QMutPy allows us to select from five quantum mutation operators:

1. QGD—Quantum Gate Deletion (equivalent to the Remove operator from Muskit)
2. QGI—Quantum Gate Insertion (equivalent to the Add operator from Muskit)
3. QGR—Quantum Gate Replacement (equivalent to the Replace operator from Muskit)
4. QMD—Quantum Measurement Deletion
5. QMI—Quantum Measurement Insertion

ID	Topic	Paper title	Tool	Year	Reference
Verification					
1	Hoare Logic	Floyd–Hoare Logic for Quantum Programs	Unavailable	2012	[9]
2	Hoare Logic	An Applied Quantum Hoare Logic	Unavailable	2019	[10]
3	Static analysis	QChecker: Detecting Bugs in Quantum Programs via Static Analysis	Available	2023	[12]
4	Static/Dynamic analysis	The Smelly Eight: An Empirical Study on the Prevalence of Code Smells in Quantum Computing	Available	2023	[51]
5	Static analysis	Quantum abstract interpretation	Unavailable	2021	[13]
6	Static analysis	Static Entanglement Analysis of Quantum Programs	Unavailable	2023	[14]
7	Static analysis	A Uniform Representation of Classical and Quantum Source Code for Static Code Analysis	Unavailable	2023	[15]
8	Static analysis	LintQ: A Static Analysis Framework for Qiskit Quantum Programs	Unusable	2023	[33]

https://github.com/Z-928/QChecker
https://github.com/jose/qsmell
https://anonymous.4open.science/r/LintQ/README.md

Validation					
9	Data generation	QuanFuzz: Fuzz Testing of Quantum Program	Unavailable	2018	[16]
10	Data generation	QDiff: Differential Testing of Quantum Software Stacks	Unusable	2021	[17]
11	Data generation	Mutation-Based Test Generation for Quantum Programs with Multi-Objective Search	Unavailable	2022	[18]
12	Oracle generation	Metamorphic testing of oracle quantum programs	Unusable	2022	[19]
13	Oracle generation	MorphQ: Metamorphic Testing of Quantum Computing Platforms	Available	2022	[20]
14	Data/Oracle generation	Application of Combinatorial Testing To Quantum Programs	Available	2021	[21, 22]
15	Data/Oracle generation	Generating Failing Test Suites for Quantum Programs With Search	Available	2021	[23, 24]
16	Data/Oracle generation	Assessing the Effectiveness of Input and Output Coverage Criteria for Testing Quantum Programs	Available	2021	[25, 26]

https://github.com/UCLA-SEAL/QDiff
https://github.com/LuisLlana/metamorphic_testing/
https://github.com/sola-st/MorphQ-Quantum-Qiskit-Testing-ICSE-23/
https://github.com/Simula-COMPLEX/qucat-paper
https://github.com/Simula-COMPLEX/qusbt-tool
https://github.com/Simula-COMPLEX/quito

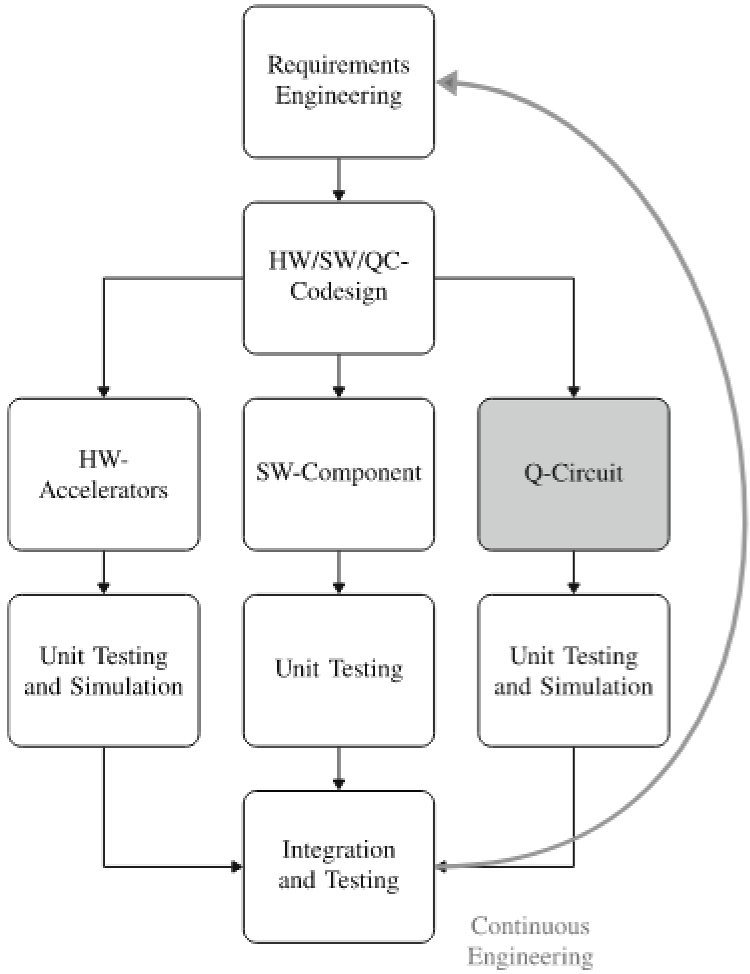


Fig. 2 Quantum software development process

Table 1 Characteristic features of different QC platforms for selected properties split into the two categories of universal and analog QC, based on [5, 6, 7, 8]. Note that no clear preference can be given due to dependencies and trade-offs between several parameters. Some QC platforms might belong to both categories

Universal QC	Number of qubits	Qubit lifetimes	Gate times	Single-qubit gate fidelities	Quantum volume
Superconducting qubits	LARGE (100+)	$\sim$ 0.5 ms	$\sim$ 10–50 ns	$\sim$ 99.9%	MEDIUM
Trapped ions	MEDIUM (<100)	$\sim 50 + s$	$\sim$ 1–50 µs	$\sim$ 99.99%	HIGHEST
Neutral atoms	LARGE	$\sim$ 1 s	$\sim$ 100 ns	$\sim$ 99%	
Photons	SMALL (<10)	N/A	$\sim$ 1 ns	$\sim$ 99.9%	
Quantum dots	SMALL	$\sim$ 1–10 s	$\sim$ 1–10 ns	$\sim$ 99%	
Semiconductors	SMALL	$\sim$ 100 µs	$\sim$ 10–20 µs	$\sim$ 99%	
Analog QC	**Number of qubits**	**Problem sizes**			
Quantum annealer	LARGE	MEDIUM			
Trapped ions	LARGE	MEDIUM			
Neutral atoms	LARGE	MEDIUM			

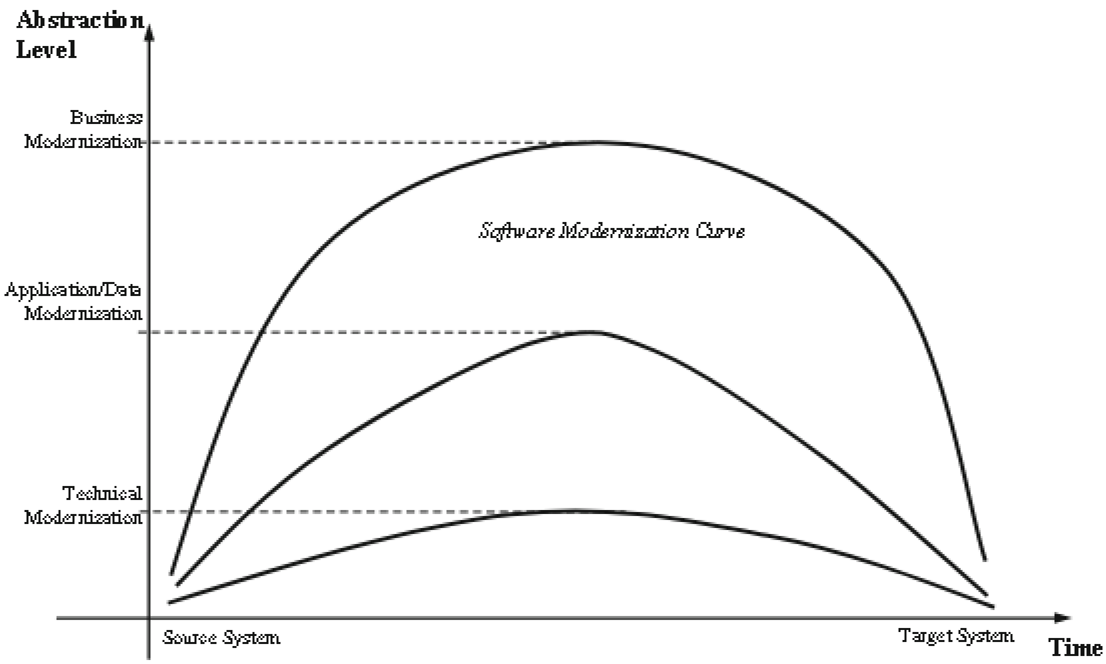


Fig. 5 Three kinds of horseshoe modernization models

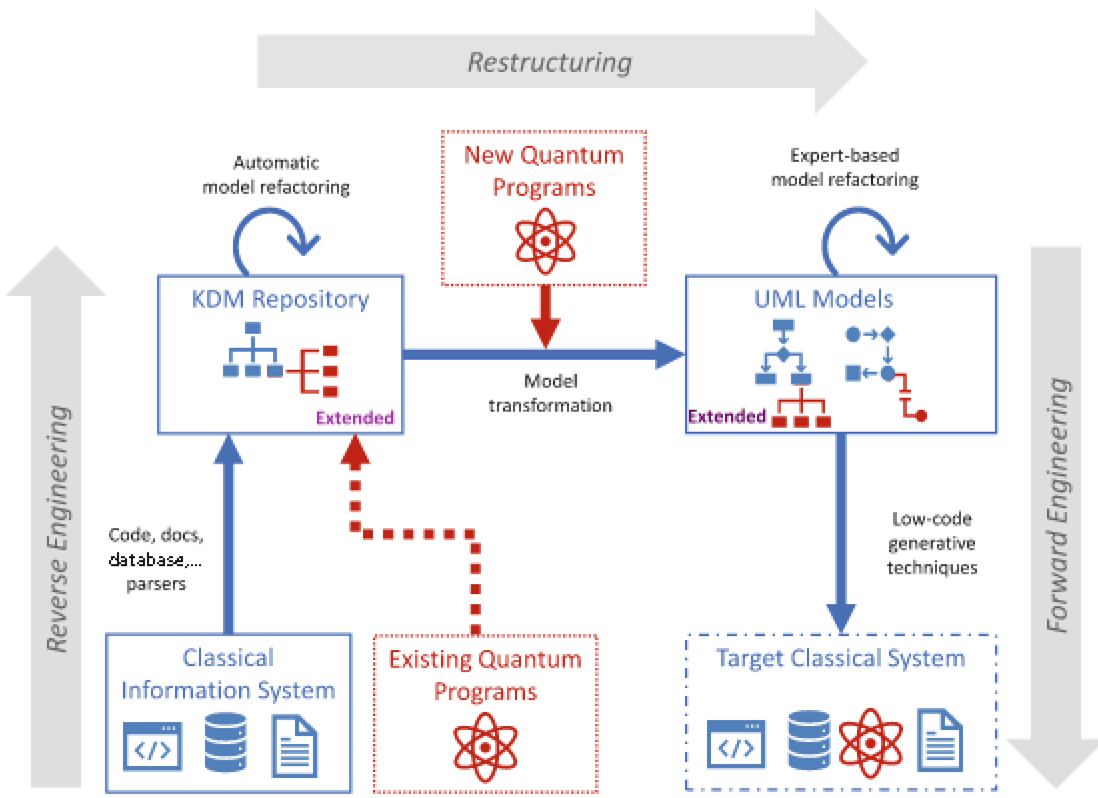


Fig. 6 Quantum software modernization approach [37]

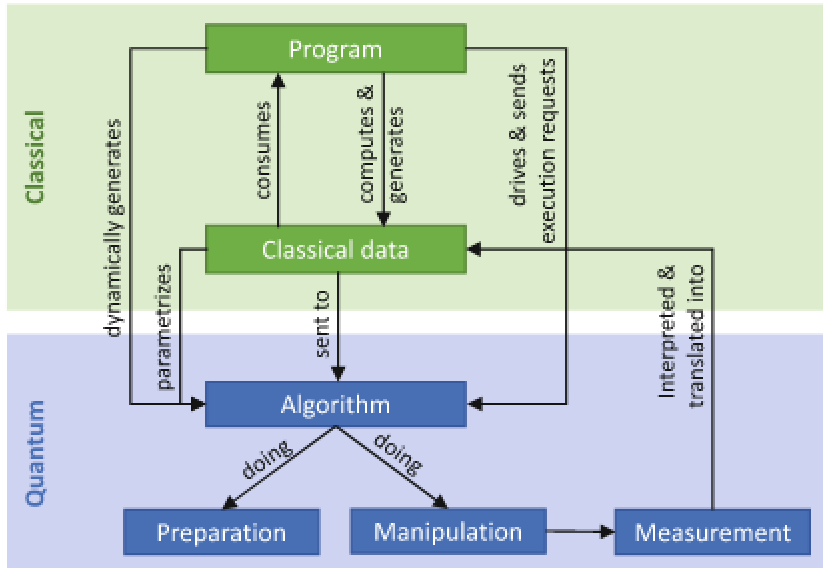
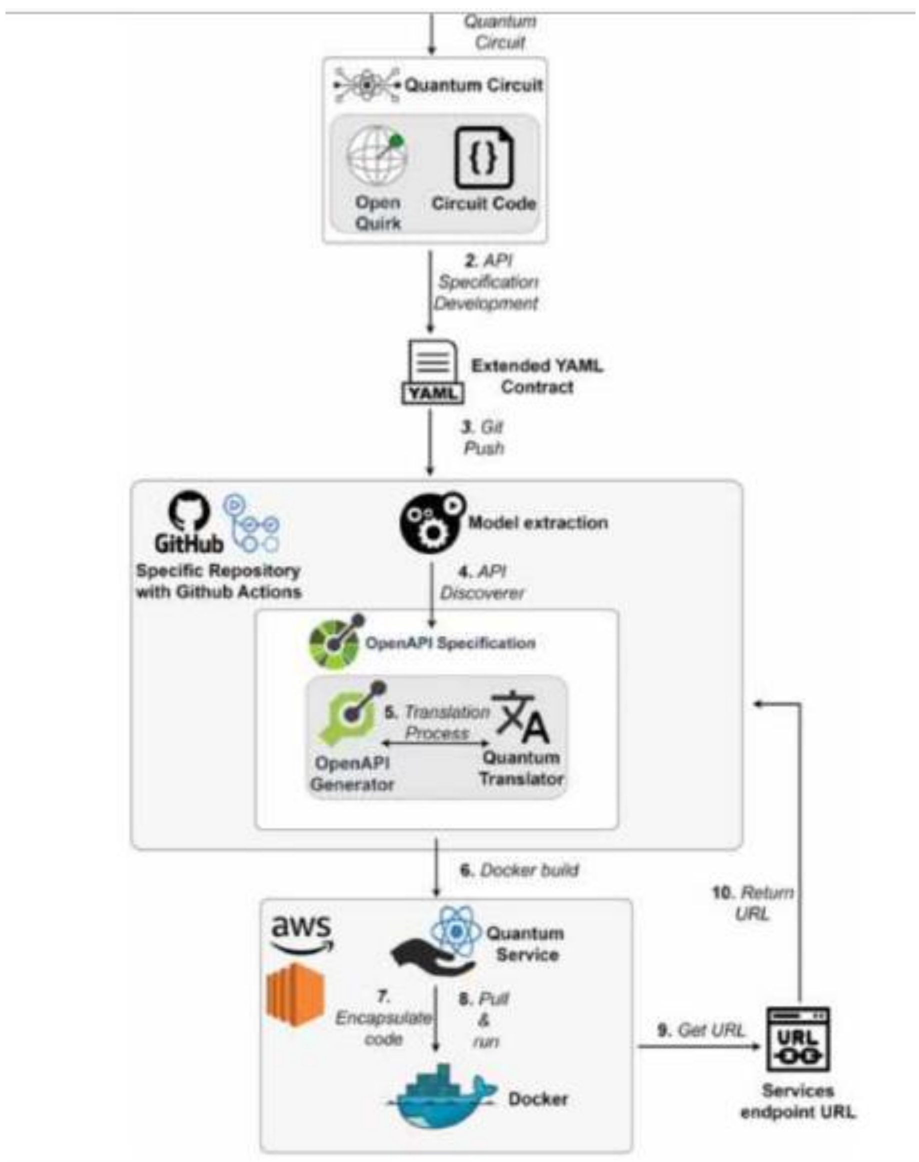


Fig. 1 Execution process of a hybrid program



Our focus:

At our startup, the key focus is the assembly and integration of the hybrid QAI software and systems. These need a special and unique model derived from individual technologies. Here is a sample case study from the book that offers good example for us to understand and apply to our framework

Fig. 1 Overview of the QHealth project

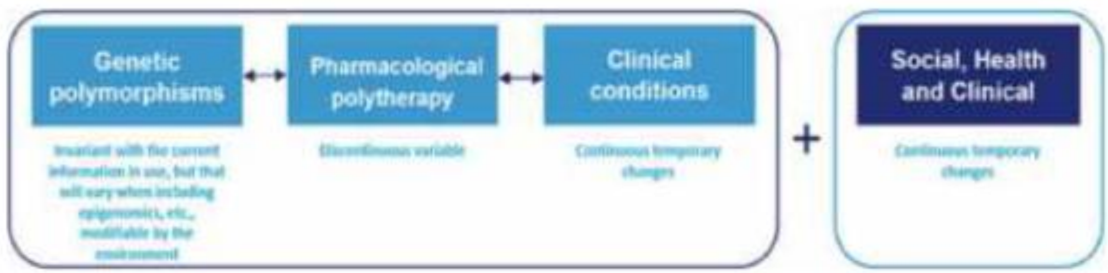


Fig. 2 Q-PGx model: QHealth's variables

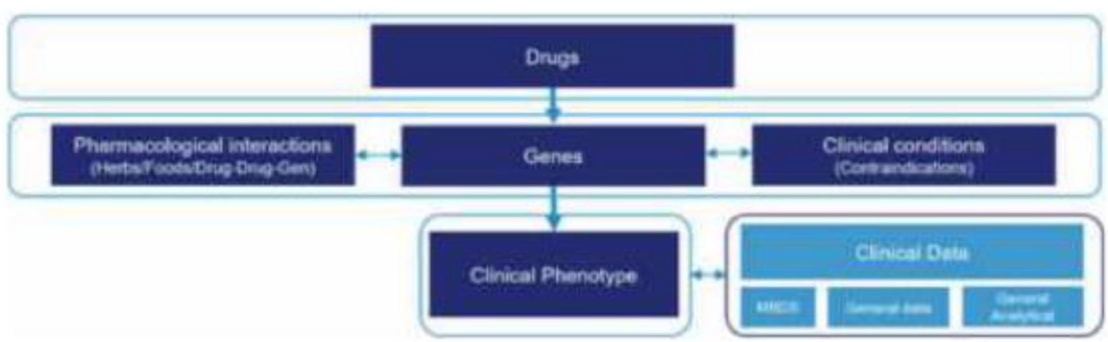


Fig. 3 Q-PGx Model: QHealth's data

Fig. 4 Relationship between a drug and the metabolizing capacity of a patient through their genotype

One of the challenges in drug prescription is the need to identify and avoid, as far as possible, interactions between drugs. When different drugs are administered to a patient, there is the possibility of adverse reactions because one of the drugs may increase or decrease the effect of another drug. This type of interaction, known as gene–drug–drug interaction, hereinafter DDI (drug–drug interaction), involves the process by which drugs are metabolized in the body

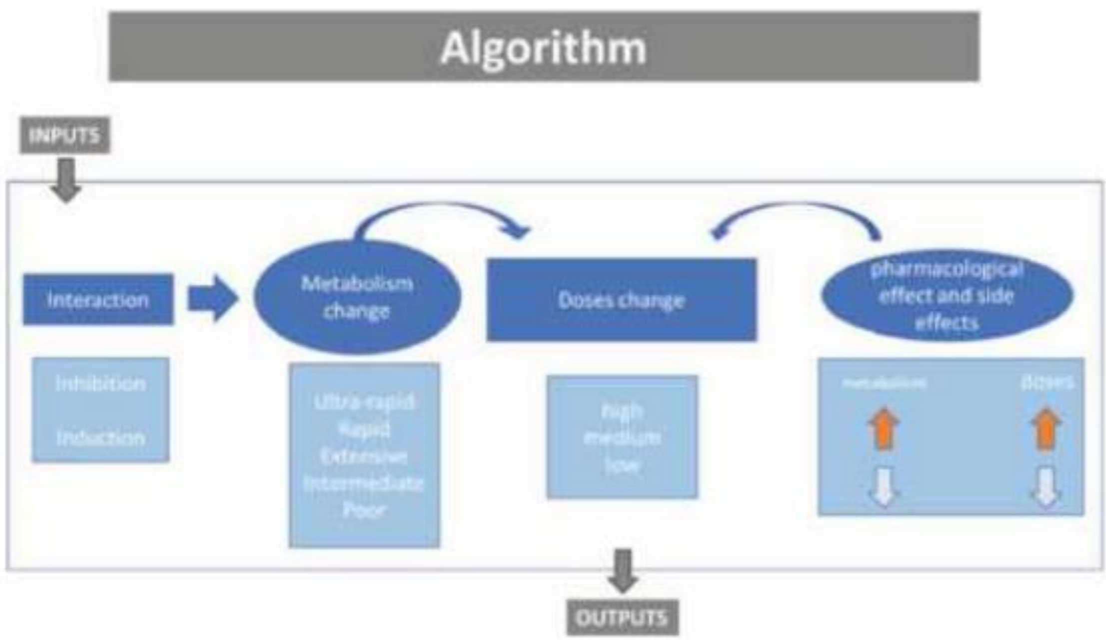


Fig. 5 QHealth's pharmacogenomic problem

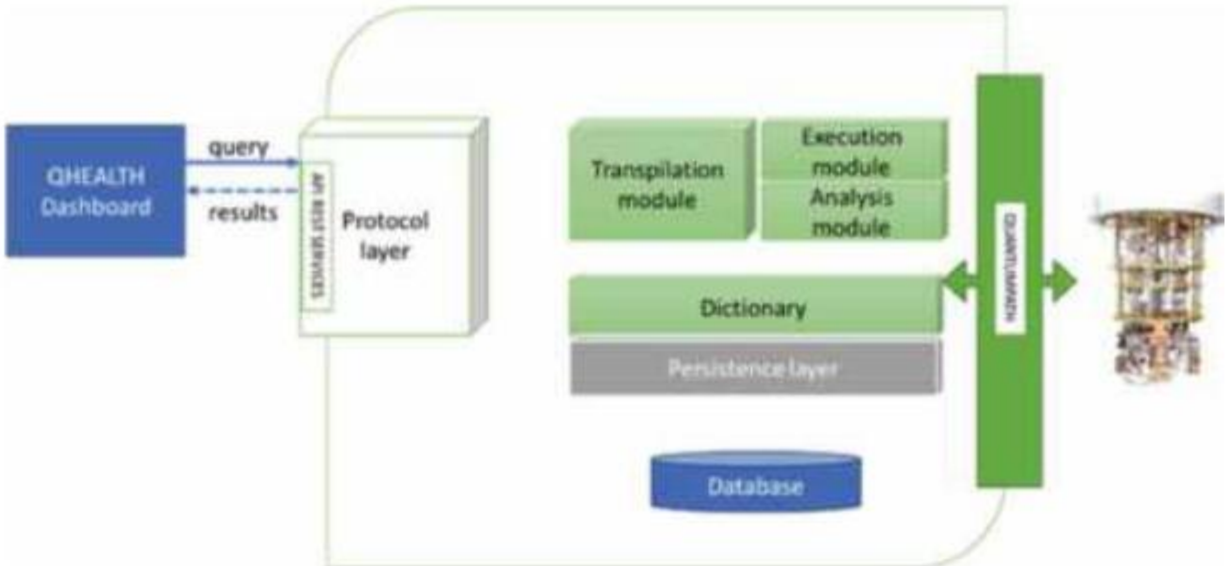


Fig. 6 System overview

Fig. 11 Dictionary module role

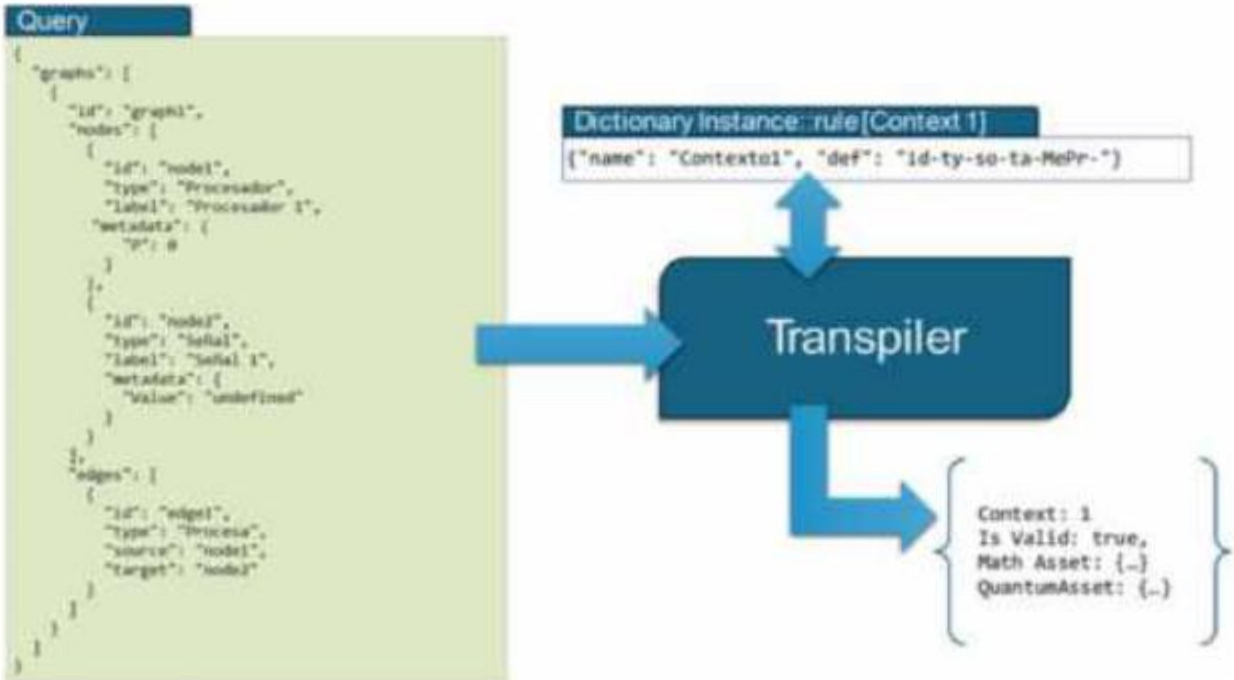


Fig. 12 Transpiler plus dictionary instance to process the query

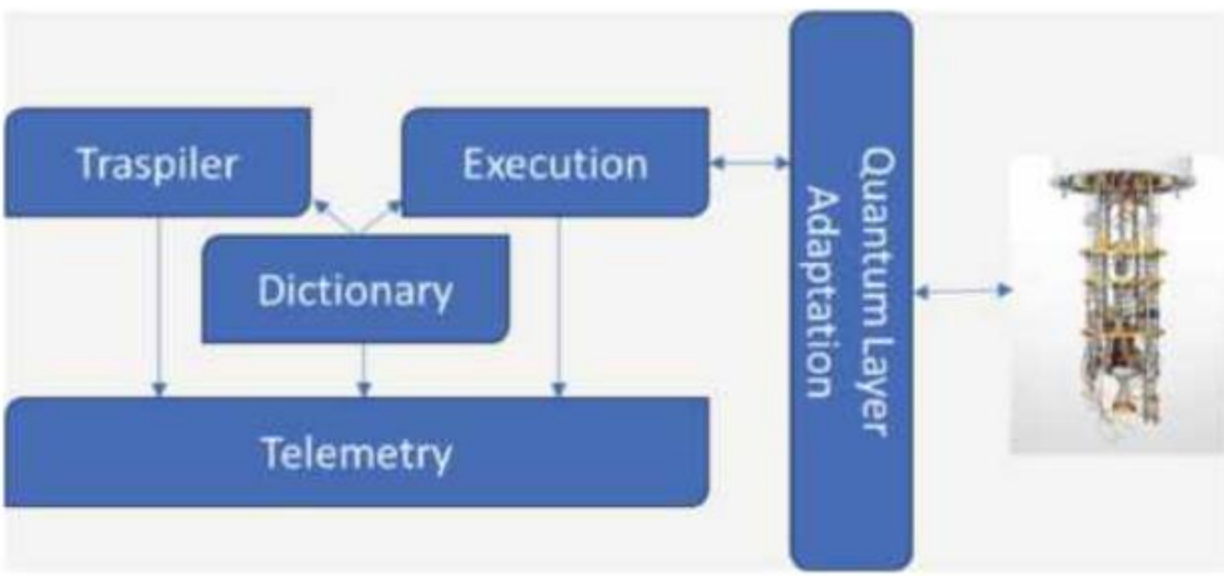


Fig. 14 Dependencies between modules