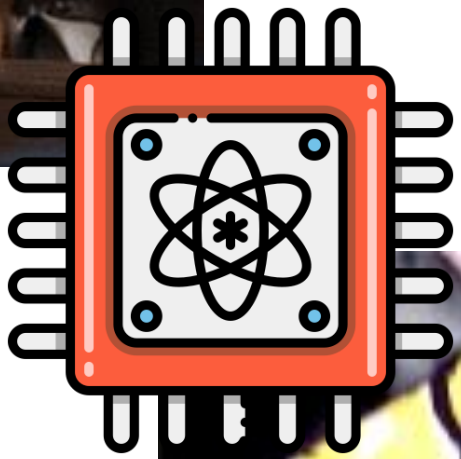
Quantum Web Services

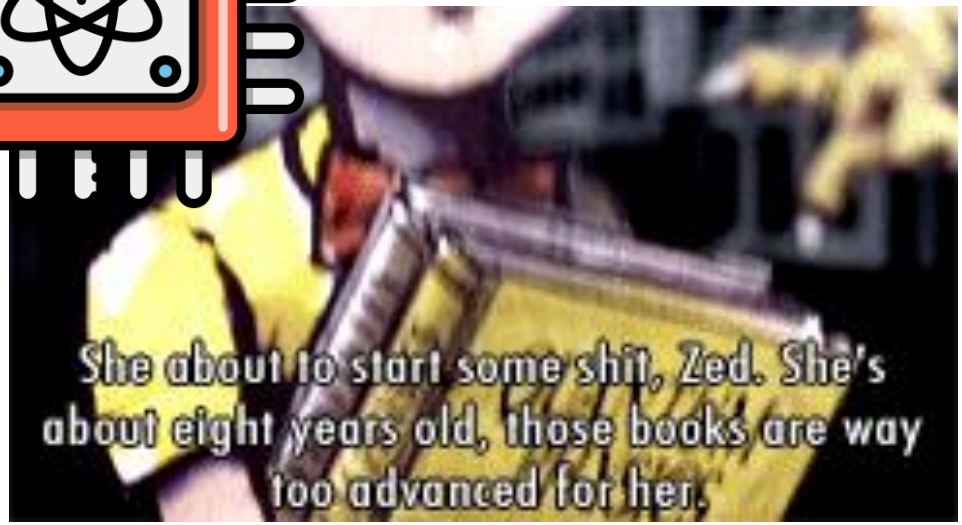
Jose Garcia-Alonso – jgaralo@unex

SPILab by Quercus SEG

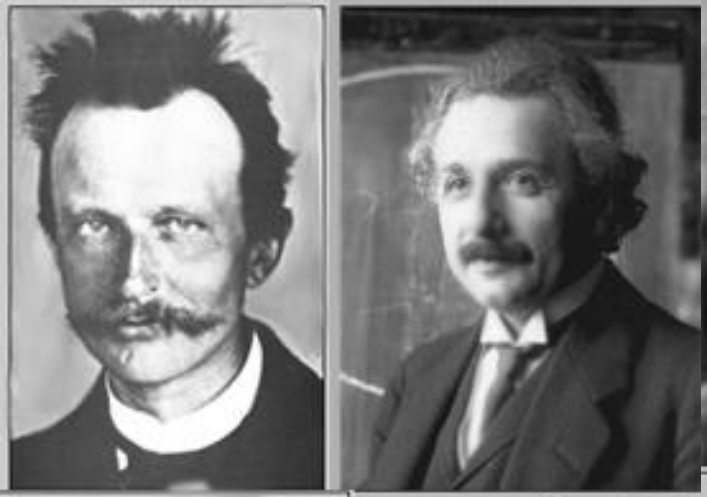
TECHNISCHE UNIVERSITÄT CHEMNITZ


Do you guys just
put the word "quantum"




She about to start some shit, Zed. She's
about eight years old, those books are way
too advanced for her.


KEEP CALM
IT'S NOT
ROCKET SCIENCE



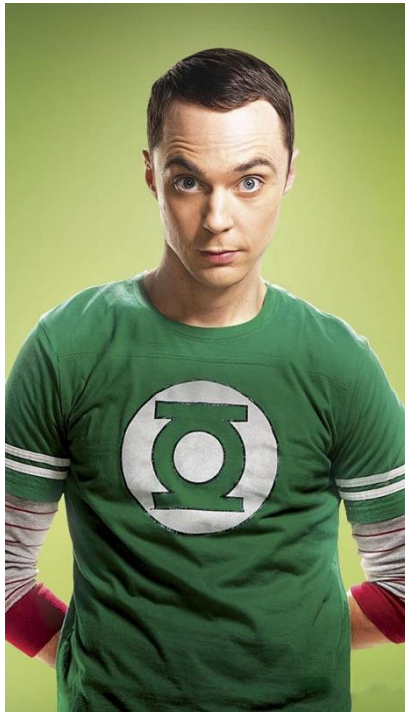







- Einstein
- Planck
- Bohr
- Feynman
- Schrödinger
- …

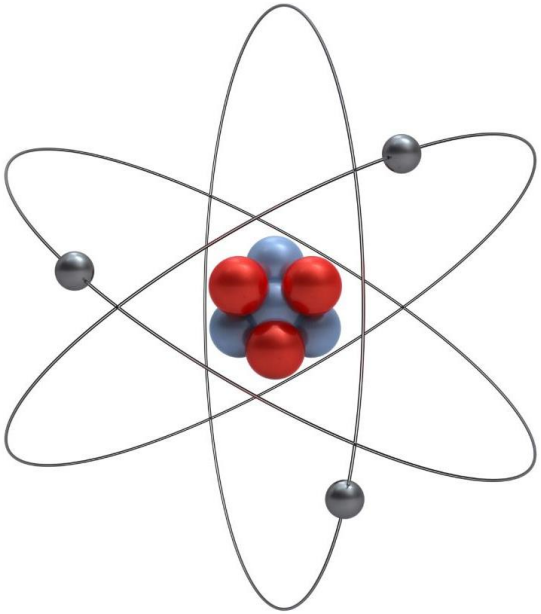
Quantum Computing



Theoretical Physics → Experimental Physics → Engineering

Quantum Computing Principles

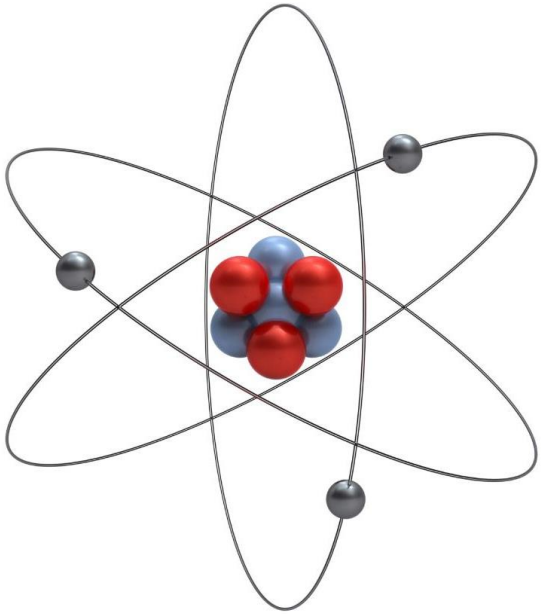
Take advantage of quantum mechanics to compute (the physical properties of nature at a subatomic scale).



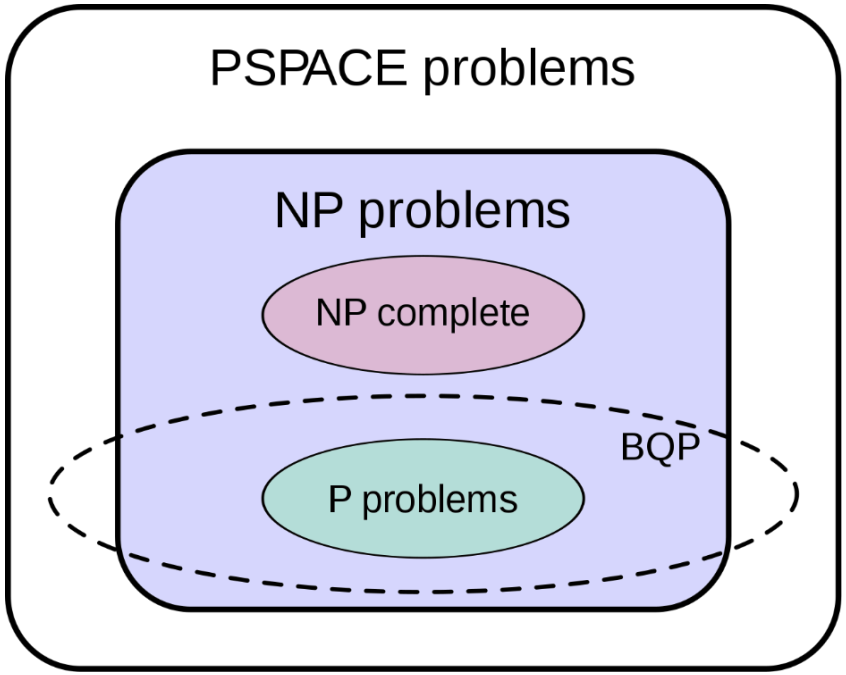
Quantum Computing Principles

Specifically:

- **Superposition**: being in different states at the same time
- **Colapse**: once observed, the system will always be in one of the posible states.
- **Entanglement**: The state of a particle cannot be described independently of the state of others



Quantum Computing – Why?



- **1998** – Los Alamos National Laboratory – Massachussetts Institute of Technology
- Primer computador cuántico
  - 2 qubits
- Resonancia magnética nuclear

IBM

Google

XANADU

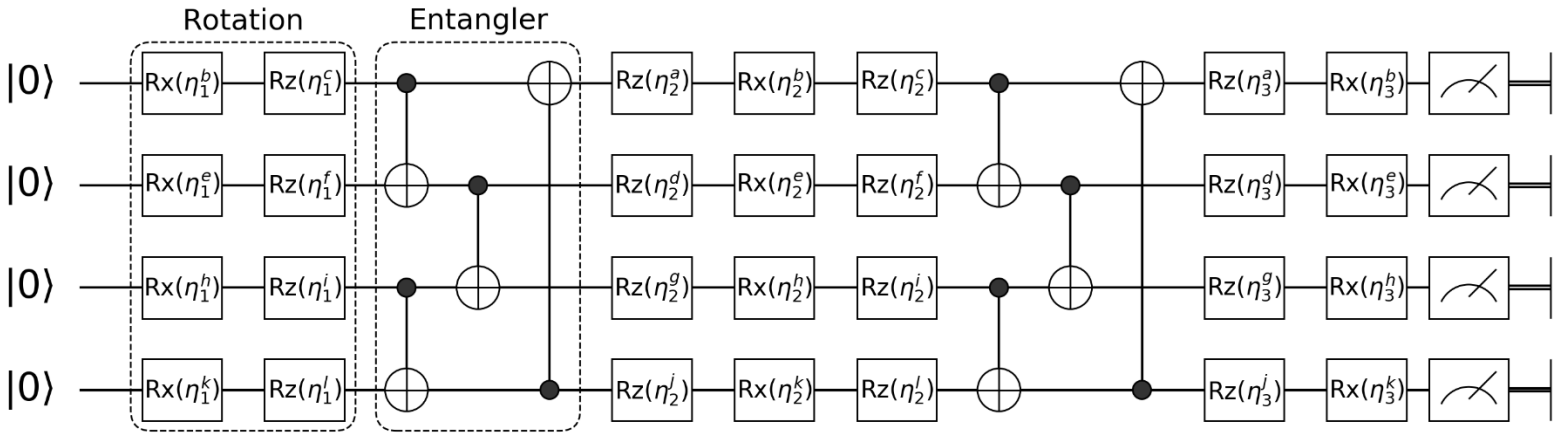
rigetti

IONQ

D:wave

The Quantum Computing Company™

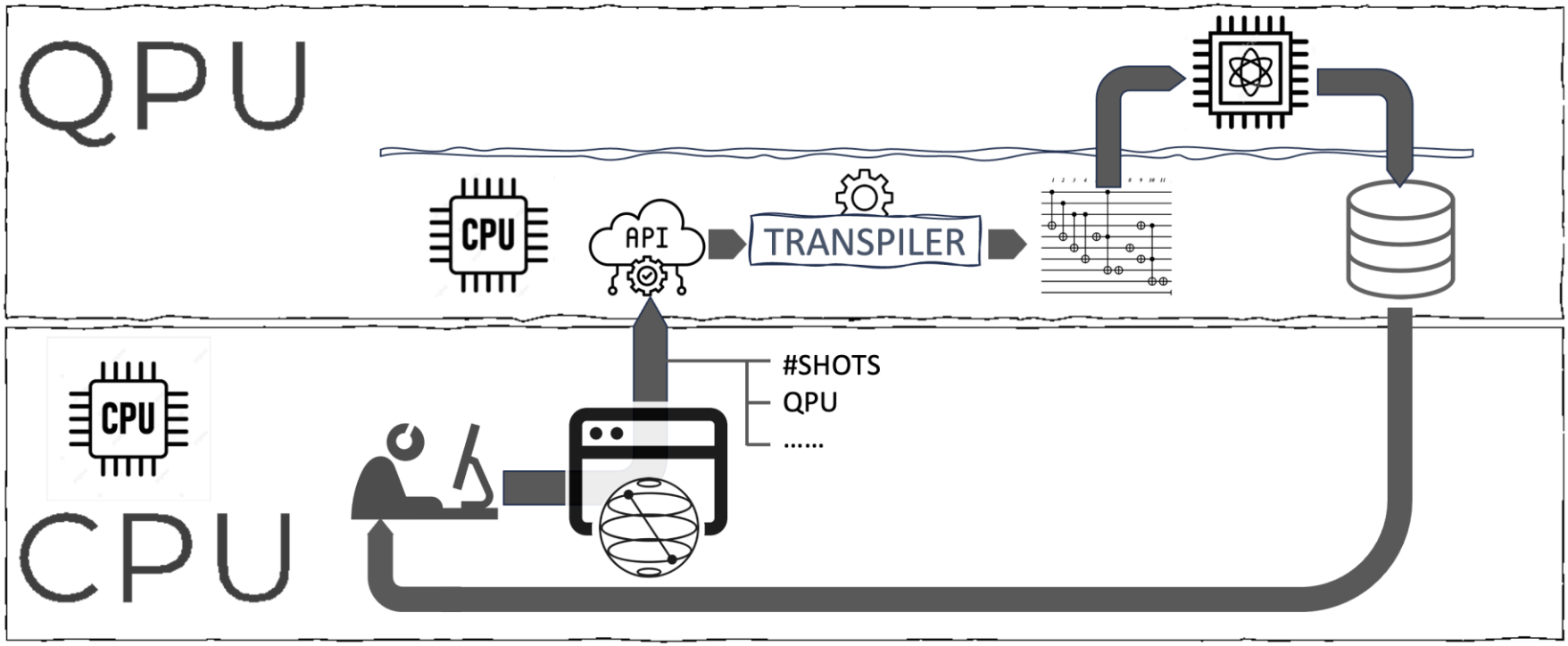
Quantum Computing – How?



Differentiable Circuit of Size 4, depth 2

```
qr0 = QuantumRegister(2, 'q')
cr0 = ClassicalRegister(2, 'c')
qc = QuantumCircuit(qr0, cr0)
qc.h(qr0[0])
qc.cx(qr0[0], qr0[1])
qc.measure(qr0, cr0)
```

Quantum Computing – How?

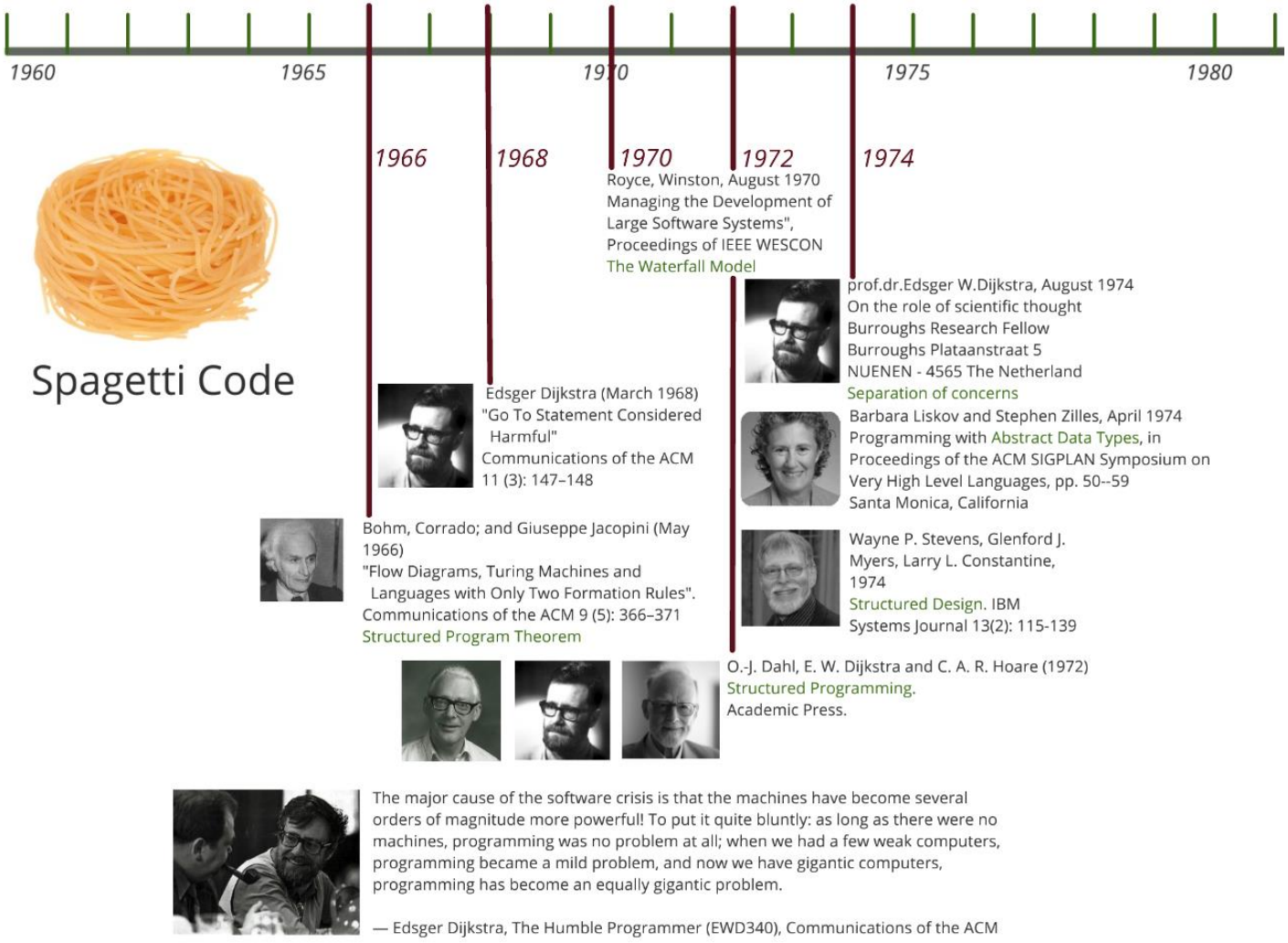


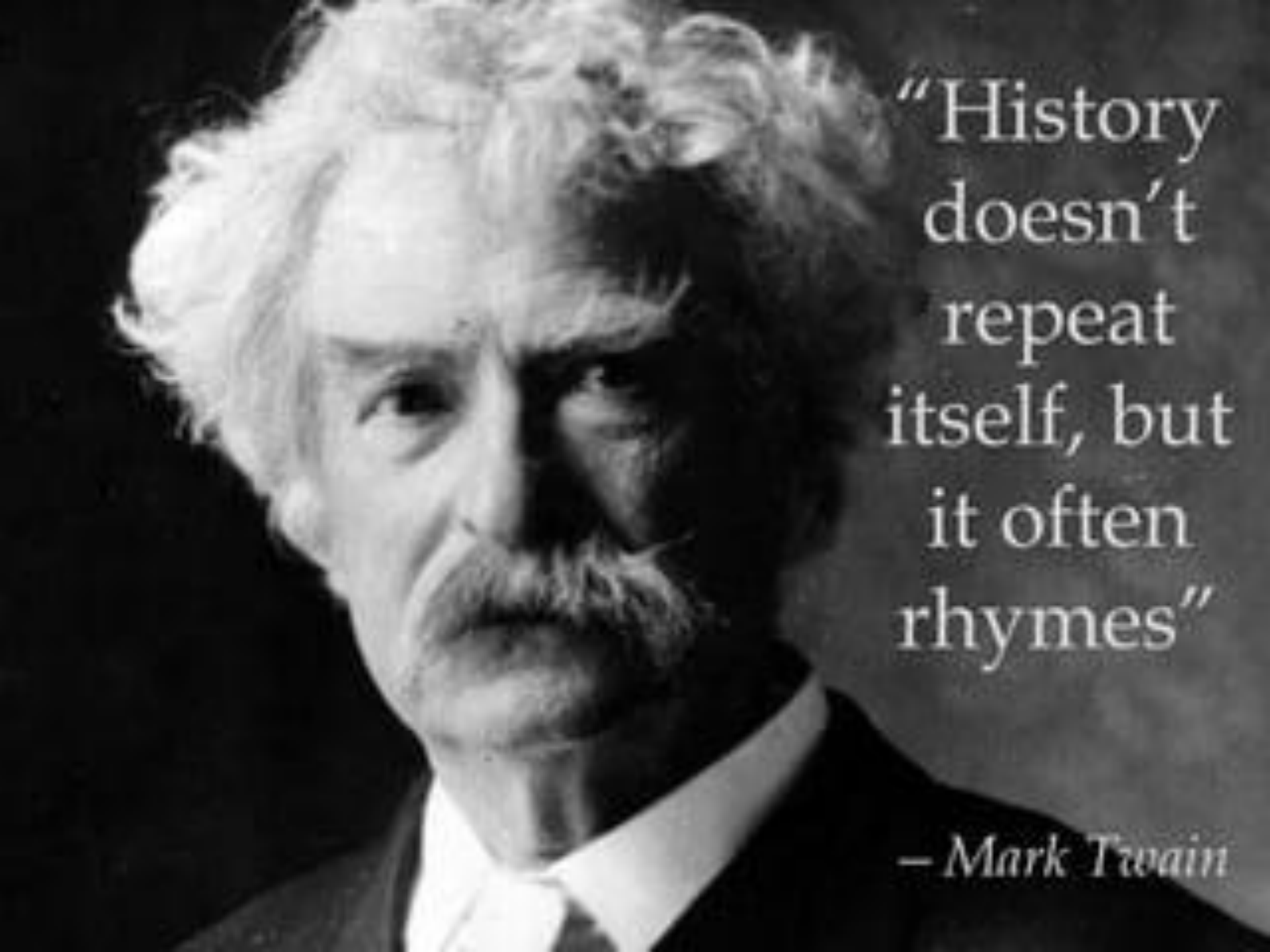
Software Engineering

Notable definitions of software engineering include:

- "The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software."—The Bureau of Labor Statistics—IEEE *Systems and software engineering – Vocabulary*[19]
- "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software."—IEEE *Standard Glossary of Software Engineering Terminology*[20]
- "An engineering discipline that is concerned with all aspects of software production."—Ian Sommerville[21]
- "The establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines."—Fritz Bauer[22]
- "A branch of computer science that deals with the design, implementation, and maintenance of complex computer programs."—Merriam-Webster[23]
- "'Software engineering' encompasses not just the act of writing code, but all of the tools and processes an organization uses to build and maintain that code over time. [...] Software engineering can be thought of as 'programming integrated over time.'"—Software Engineering at Google[24]

Software Engineering




"History doesn't repeat itself, but it often rhymes"
– Mark Twain

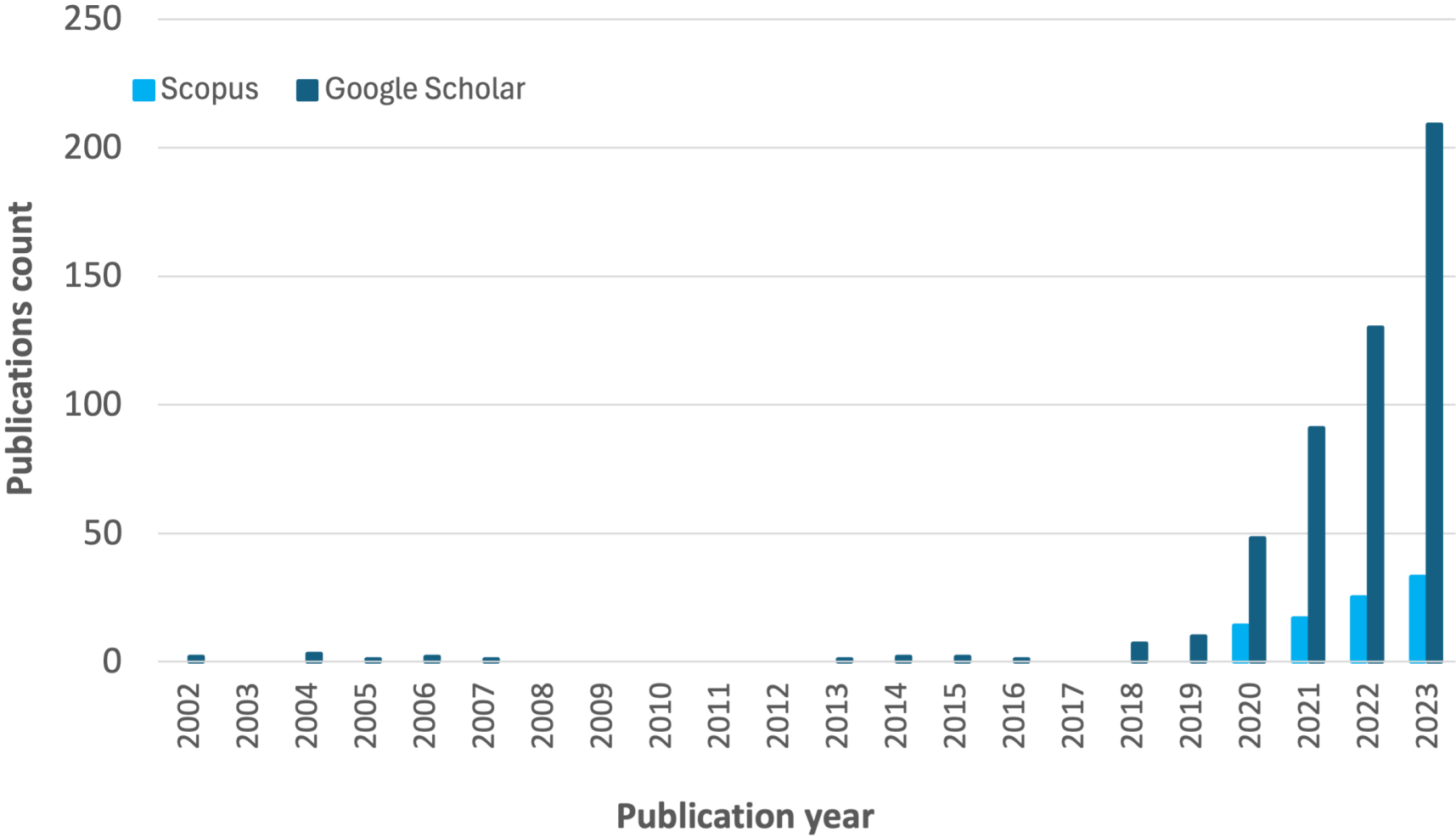
Quantum Software Engineering

"the use of sound engineering principles for the development, operation, and maintenance of quantum software"

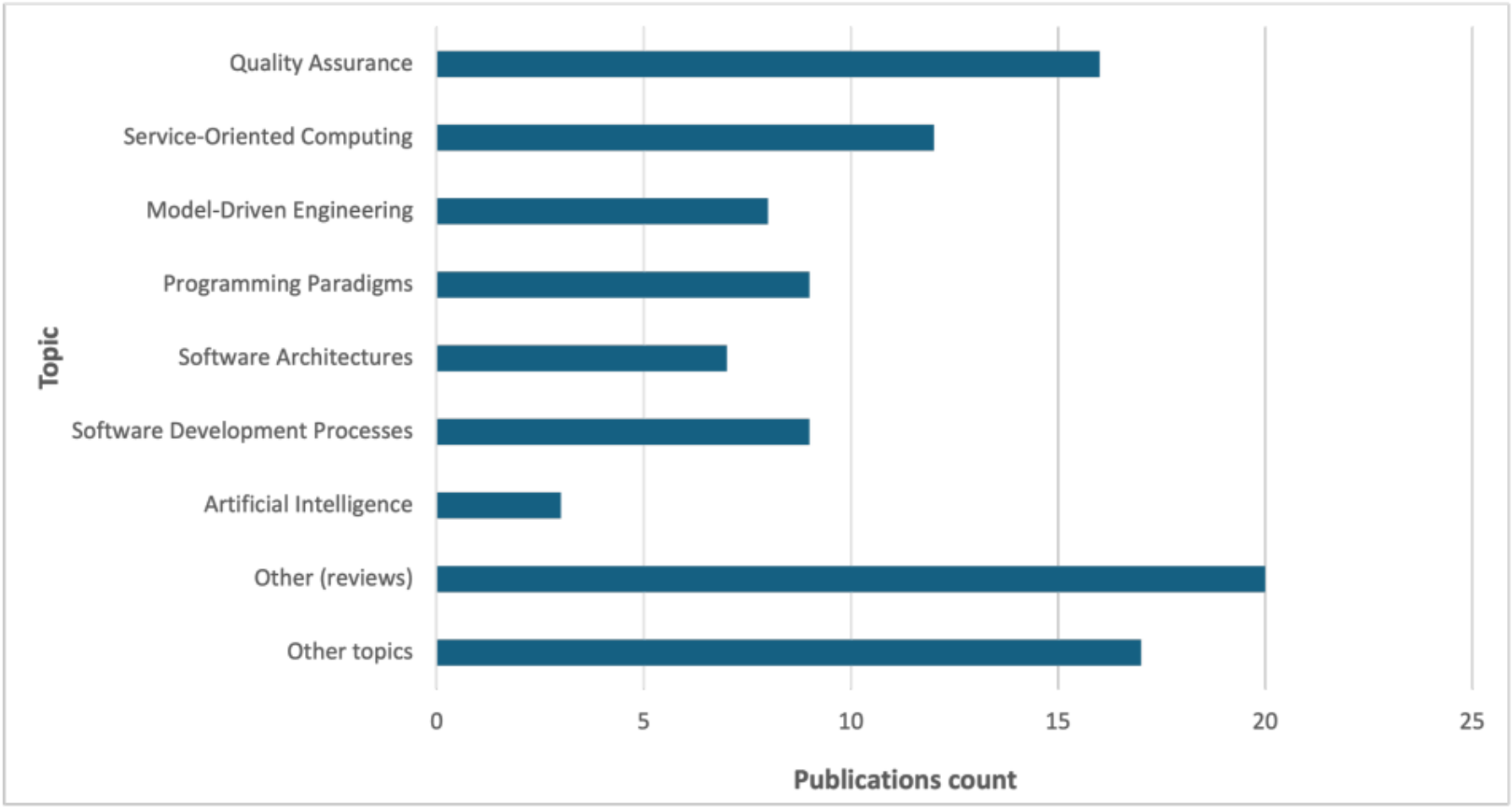
Jianjun Zhao. 2020. Quantum Software Engineering: Landscapes and Horizons. *arXiv* (jul 2020). https://doi.org/10.48550/arXiv.2007.07047

One of the main challenges in QSE is to translate the knowledge and practices from classical software engineering to the quantum domain while also developing new methodologies tailored to match the unique requirements of quantum computing

Quantum Software Engineering

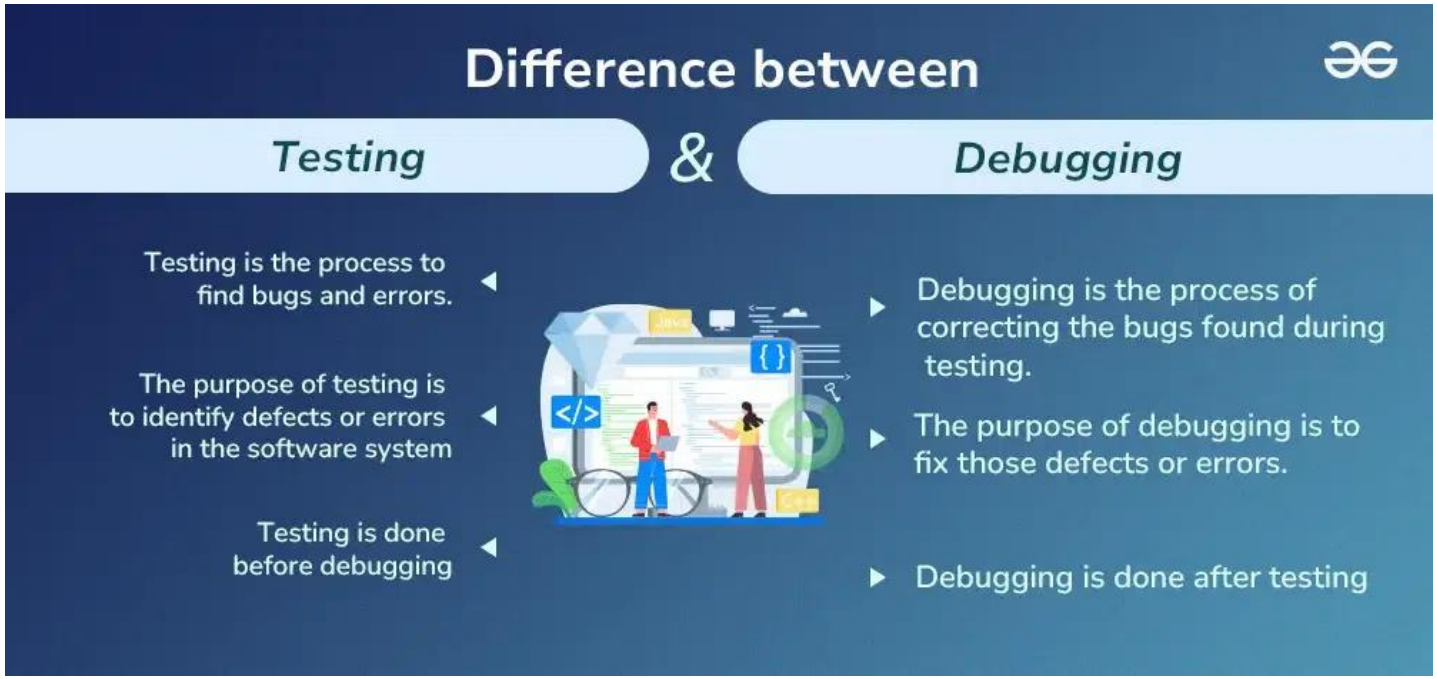


Quantum Software Engineering



Quantum Software Engineering

- QSE – Active areas
  - Quality Assurance: Testing and Debugging

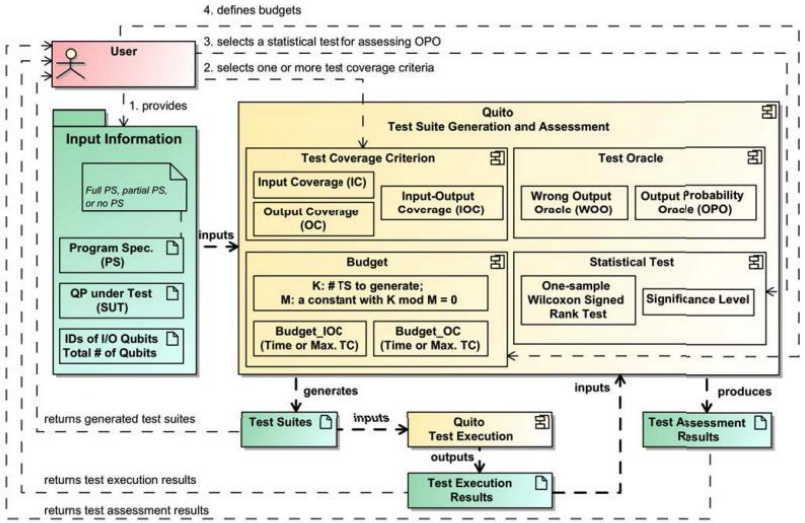


Quantum Software Engineering

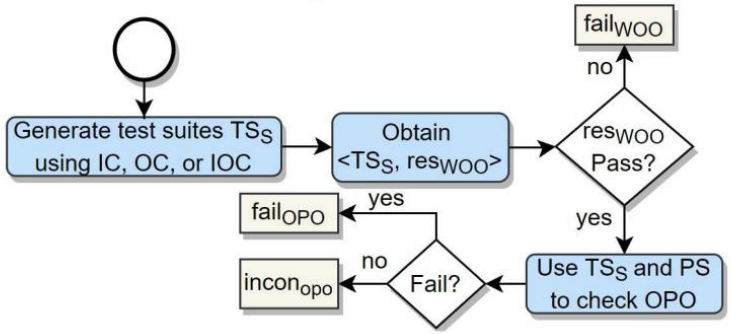
- QSE – Active areas
  - Quality Assurance: Testing and Debugging
    - Quantum software testing is about assessing the correctness of quantum software in delivering their intended functionalities
    - Debugging is about observing failures in quantum software that need to be diagnosed and fixed
    - Especially when facing challenges such as quantum noise, limited quantum hardware and simulator resources, and limited observability

Quantum Software Engineering

- QSE – Active areas
  - Quality Assurance: Testing and Debugging – Coverage



(a) Overview



Quito: a Coverage-Guided Test Generator for Quantum Programs

Xinyi Wang
Nanjing University of Aeronautics and Astronautics
Nanjing, China
wangxinyi125@nuaa.edu.cn

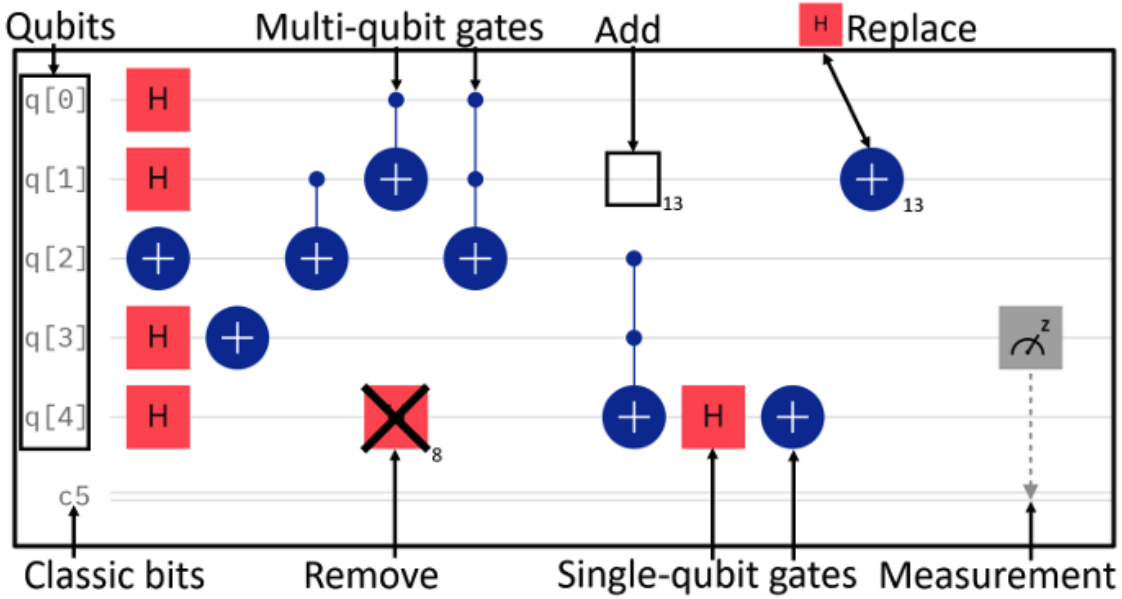
Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
arcaini@nii.ac.jp

Tao Yue
Nanjing University of Aeronautics and Astronautics, China
Simula Research Laboratory, Norway
taoyue@ieee.org

Shaukat Ali
Simula Research Laboratory
Fornebu, Norway
shaukat@simula.no

Quantum Software Engineering

- QSE – Active areas
  - Quality Assurance: Testing and Debugging –



WHICH QUANTUM CIRCUIT MUTANTS SHALL BE USED?
EMPIRICAL EVALUATION OF QUANTUM CIRCUIT MUTATIONS

Eñaut Mendiluze Usandizaga
Simula Research Laboratory
Oslo, Norway
enaut@simula.no

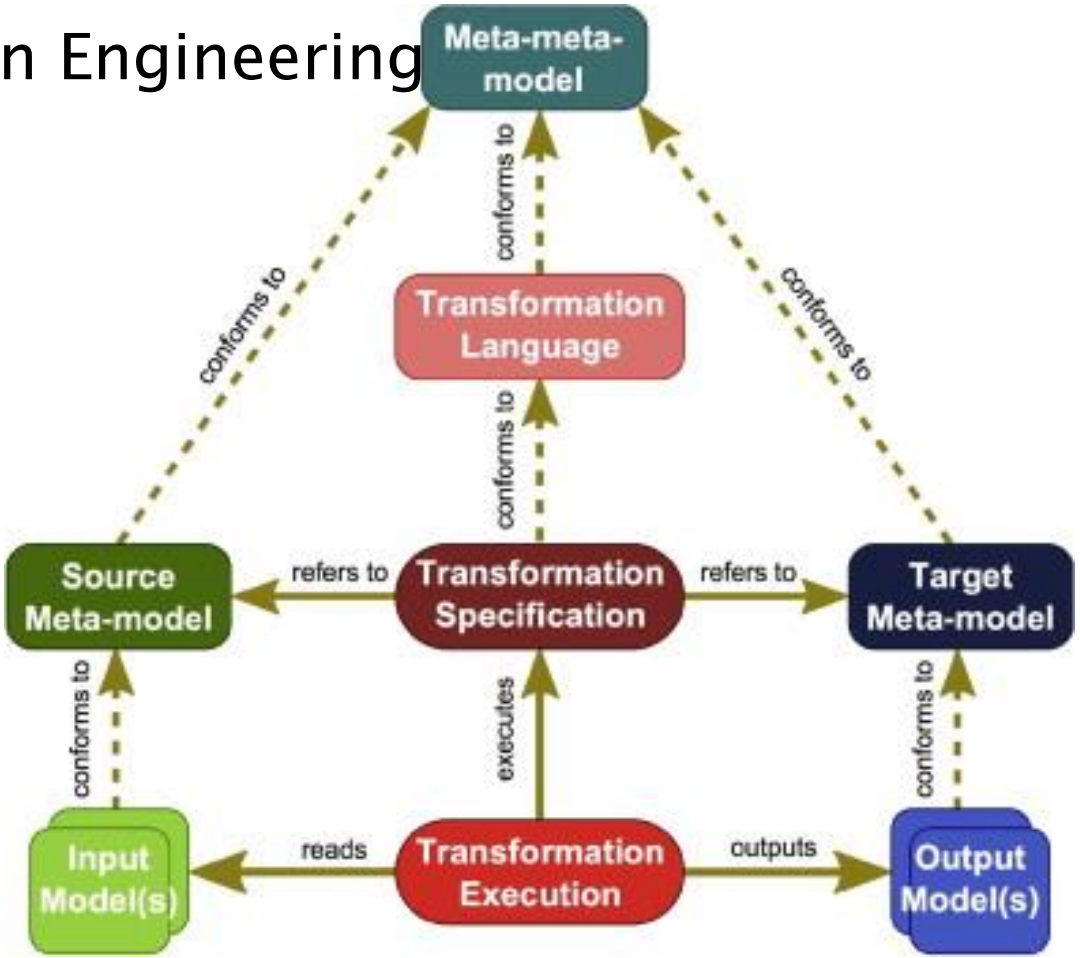
Tao Yue
Simula Research Laboratory
Oslo, Norway
taoyue@gmail.com

Paolo Arcaini
National Institute of Informatics
Tokio, Japan
arcaini@nii.ac.jp

Shaukat Ali
Simula Research Laboratory
Oslo Metropolitan University
Oslo, Norway
shaukat@simula.no

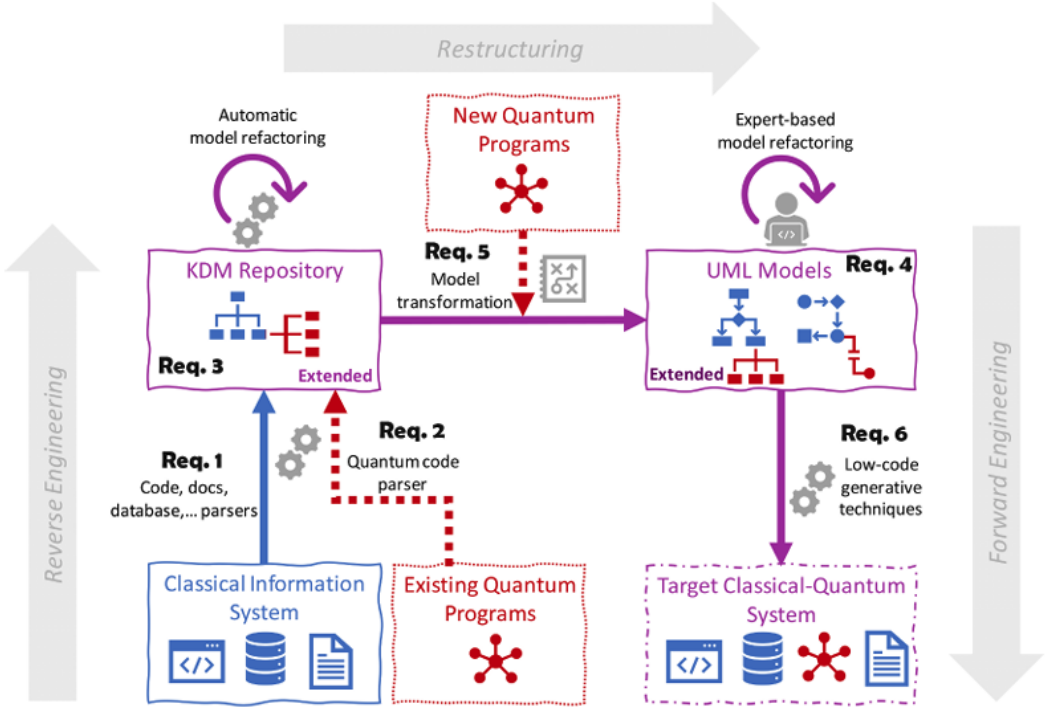
Quantum Software Engineering

- QSE – Active areas
  - Model-Driven Engineering



Quantum Software Engineering

- QSE – Active areas
  - Model-Driven Engineering – Modernization



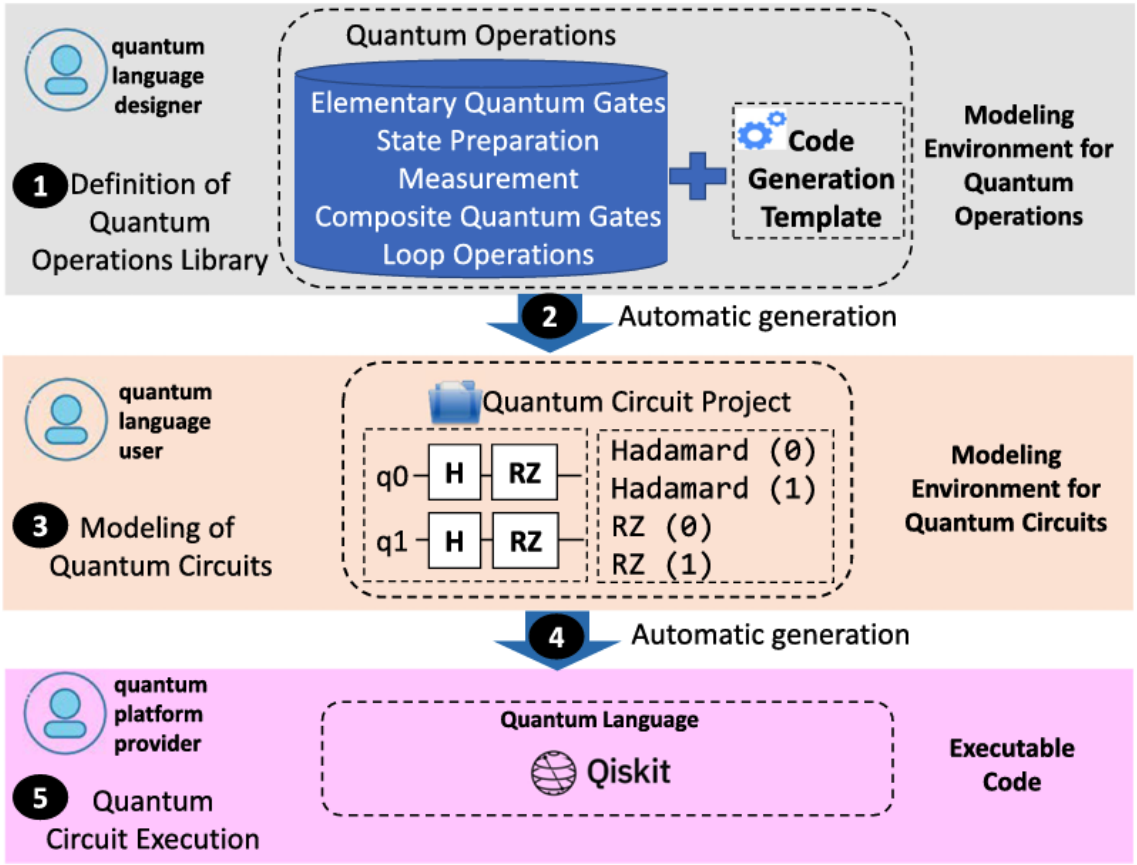
Software modernization to embrace quantum technology

Ricardo Pérez-Castillo [a,b,*], Manuel A. Serrano [b], Mario Piattini [b]

[a] Faculty of IT & Social Sciences, University of Castilla-La Mancha, Av. Real Fábrica de Sedas, s/n, 45600, Talavera de la Reina, Spain
[b] Information Technology & Systems Institute, University of Castilla-La Mancha, Paseo de la Universidad, 4, 13071, Ciudad Real, Spain, University of Castilla-La Mancha

Quantum Software Engineering

- QSE – Active areas
  - Model-Driven Engineering – Composition

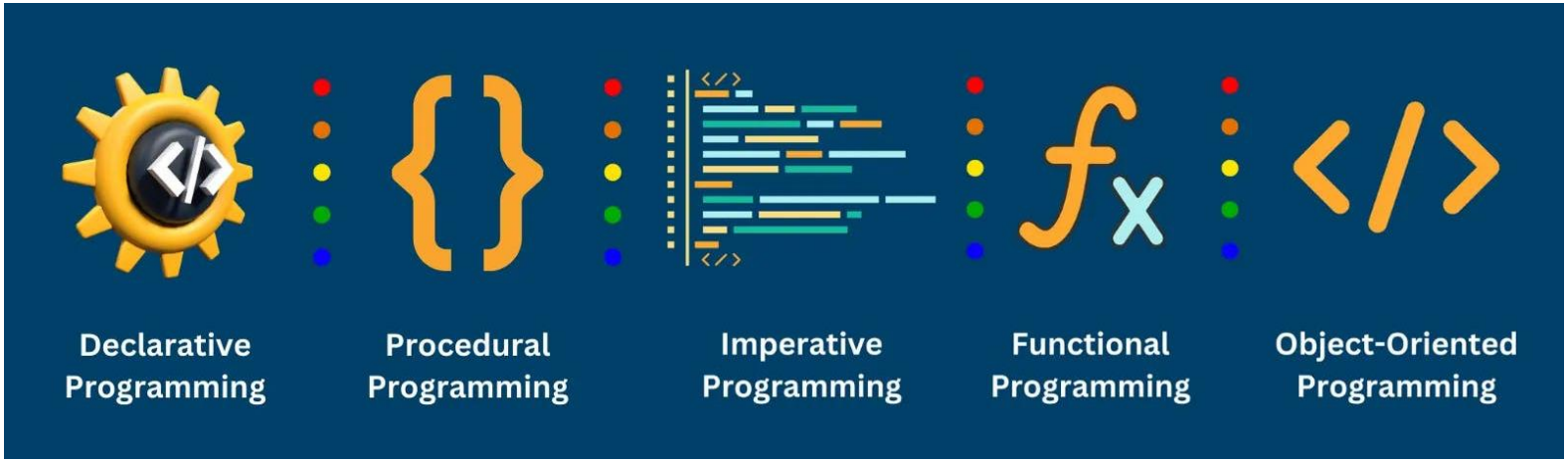


A Model-Driven Framework for Composition-Based Quantum Circuit Design

FELIX GEMEINHARDT, Business Informatics - Software Engineering, Johannes Kepler Universität Linz, Linz, Austria
ANTONIO GARMENDIA, Business Informatics - Software Engineering, Johannes Kepler Universitat Linz, Linz, Austria and Universidad Autonoma de Madrid, Madrid, Spain
MANUEL WIMMER, Business Informatics - Software Engineering, Johannes Kepler Universität Linz, Linz, Austria
ROBERT WILLE, Technical University of Munich, Munich, Germany

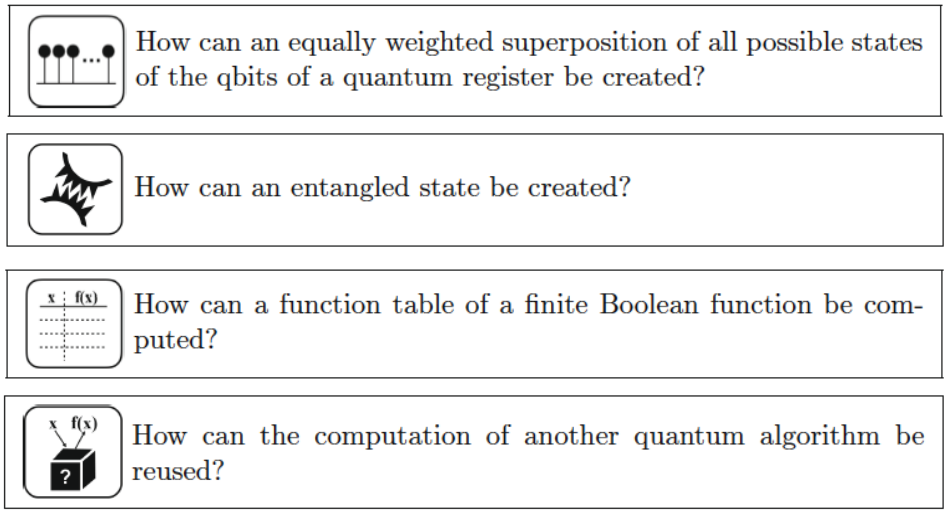
Quantum Software Engineering

- QSE – Active areas
  - Programming Paradigms



Quantum Software Engineering

- QSE – Active areas
  - Programming Paradigms – Pattern Language

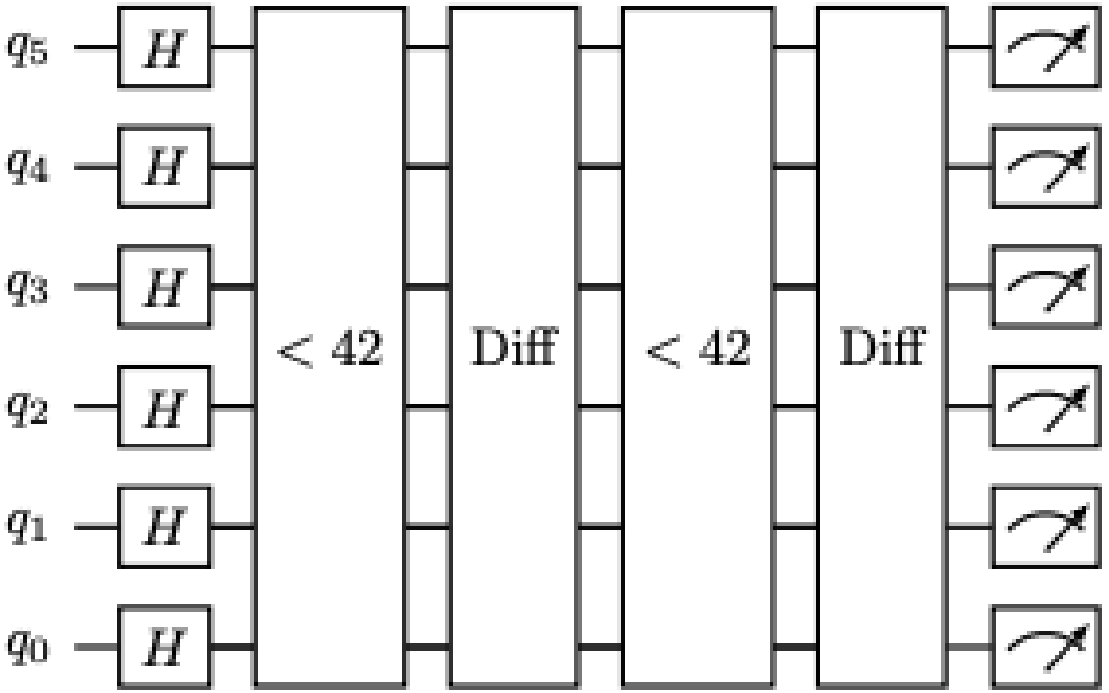


Towards a Pattern Language for Quantum Algorithms

Frank Leymann
IAAS, University of Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
Frank.Leymann@iaas.uni-stuttgart.de

Quantum Software Engineering

- QSE – Active areas
  - Programming Paradigms – Oracles



Automatic Generation of an Efficient Less-Than Oracle for Quantum Amplitude Amplification

Javier Sanchez-Rivero
COMPUTAEX
Cáceres, Spain
javier.sanchez@cenits.es

Daniel Talaván
COMPUTAEX
Cáceres, Spain
daniel.talavan@cenits.es

Jose Garcia-Alonso
University of Extremadura
Cáceres, Spain
jgaralo@unex.es

Antonio Ruiz-Cortés
SCORE Lab, I3US Institute, Universidad de Sevilla
Sevilla, Spain
aruiz@us.es

Juan Manuel Murillo
COMPUTAEX and University of Extremadura
Cáceres, Spain
juan.murillo@cenits.es

Quantum Software Engineering

- QSE – Active areas
  - Programming Paradigms – Types

```
qbit b = 0 || 1;
qbit[] bArr = qbit.zeros(4);
qint i = 15; // basis state
qint j = 1 || 30 || 160;
qint k = qint.all(); // equal superposition of ALL
↪  int values
qfloat f = 2.5 || 3.5;
qcomplex z = (0.6 + 0.8i) || (sqrt(1/2) +
↪  sqrt(1/2)i);
qchar c = 'A' || 'B' || 'X' || 'Y';
qstring s = "Hello" || "World";
qbool tf = true || false;
qref r = &i || &j;
```

Quantum types: going beyond qubits and quantum gates

Tamás Varga
Constructor Institute Schaffhausen
Schaffhausen, Switzerland
tamas.varga@constructor.org

Yaiza Aragonés-Soria
Constructor Institute Schaffhausen
Schaffhausen, Switzerland
yaiza.aragonessoria@constructor.org

Manuel Oriol
Constructor Institute Schaffhausen
Schaffhausen, Switzerland
Constructor University Bremen
Bremen, Germany
mo@constructor.org

Quantum Web Engineering

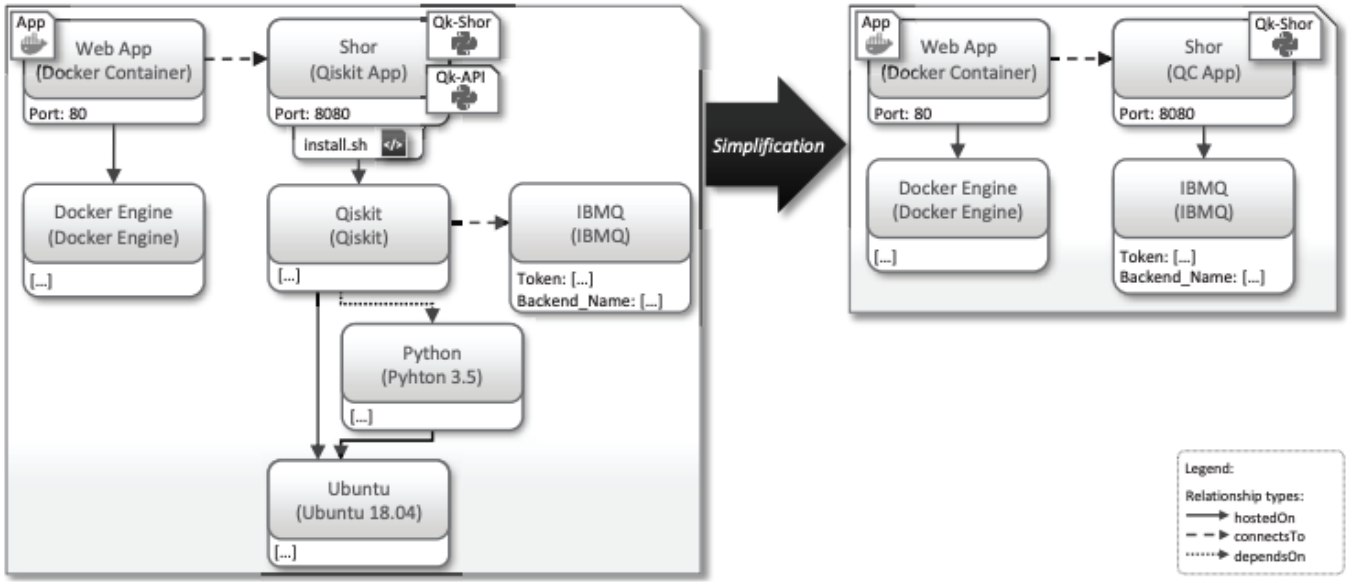
- Access to Quantum Computers through the cloud
  - Using similar techniques to traditional web Engineering
- Quantum Software will be hybrid
  - (for the foreseable future)
  - Best known way to conect distributed pieces of software running on heterogeneous hardware?
    - Web Engineering and web services

Quantum Web Engineering

- Service-Oriented Computing
  - Paradigm to support the development of interoperable, distributed applications where services are regarded as autonomous entities
  - Provisioning and aggregation of basic services, composition of complex systems through the coordination of services, management and monitoring of services deployed in the cloud or on-premise,…
  - Quantum hardware and software offered in the cloud allow users to access quantum computing resources over the internet without the need to own quantum computing hardware

Quantum Web Engineering

- Service-Oriented Computing – TOSCA

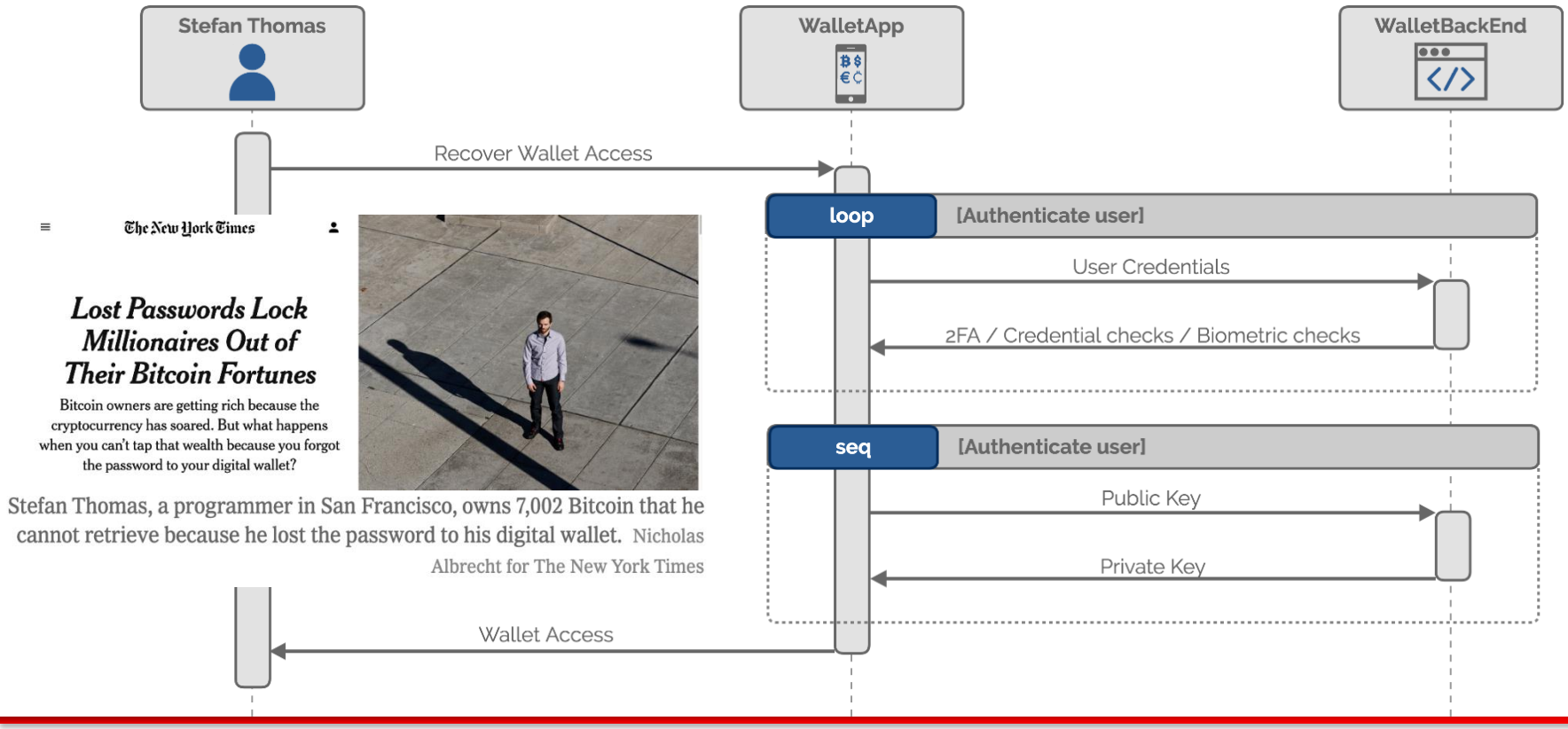


TOSCA4QC: Two Modeling Styles for TOSCA to Automate the Deployment and Orchestration of Quantum Applications

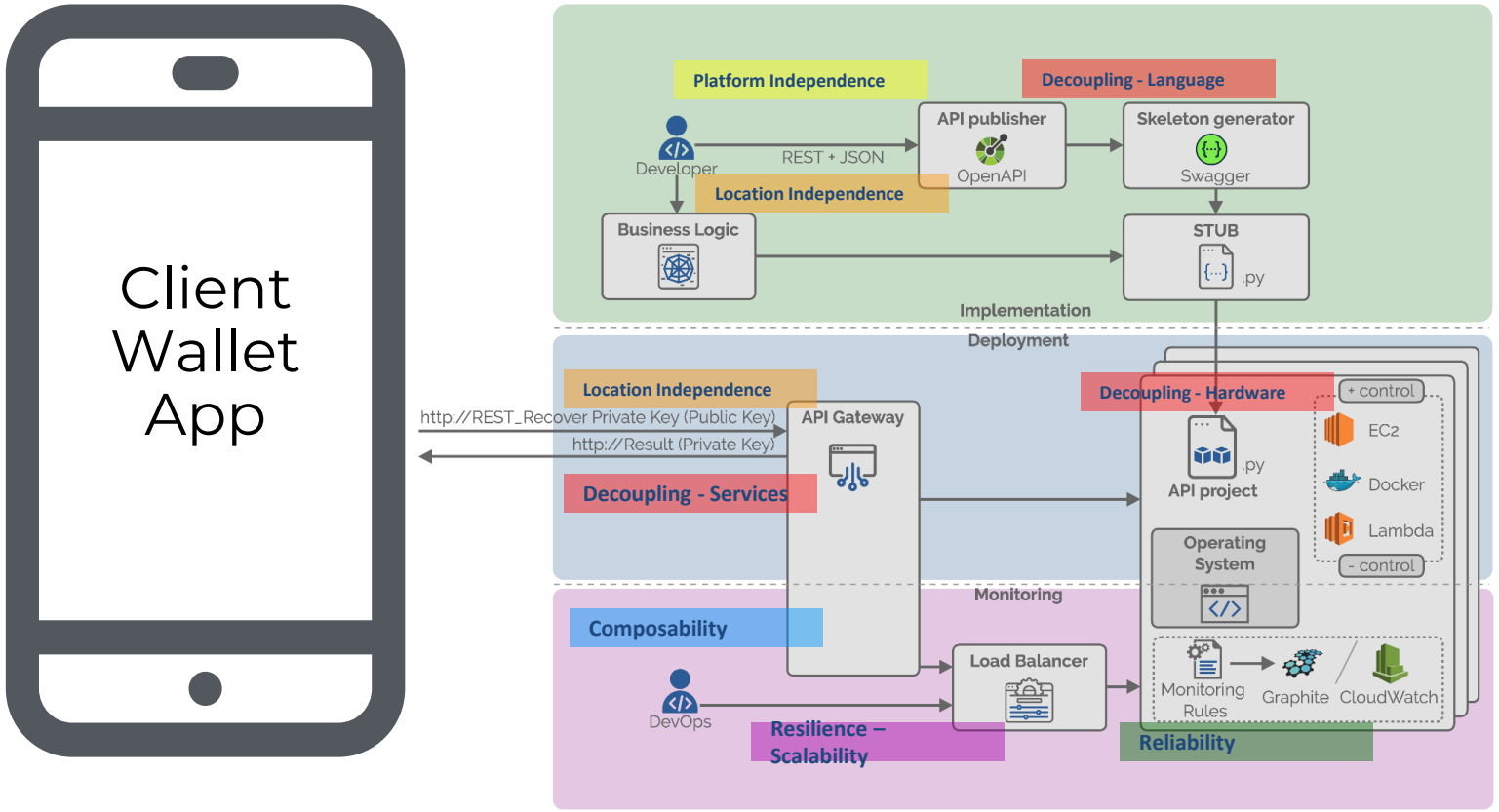
Karoline Wild, Uwe Breitenbücher, Lukas Harzenetter, Frank Leymann, Daniel Vietz, and Michael Zimmermann
Institute of Architecture of Application Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
{wild, breitenbuecher, harzenetter, leymann, vietz, zimmermann}@iaas.uni-stuttgart.de

Quantum Web Engineering

Look at a good classic service implementation

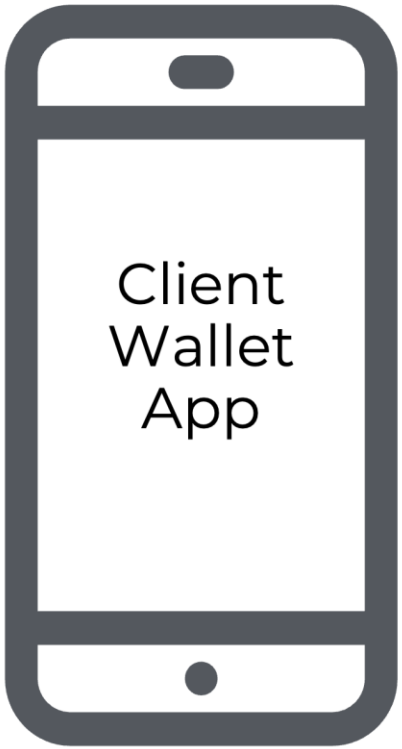


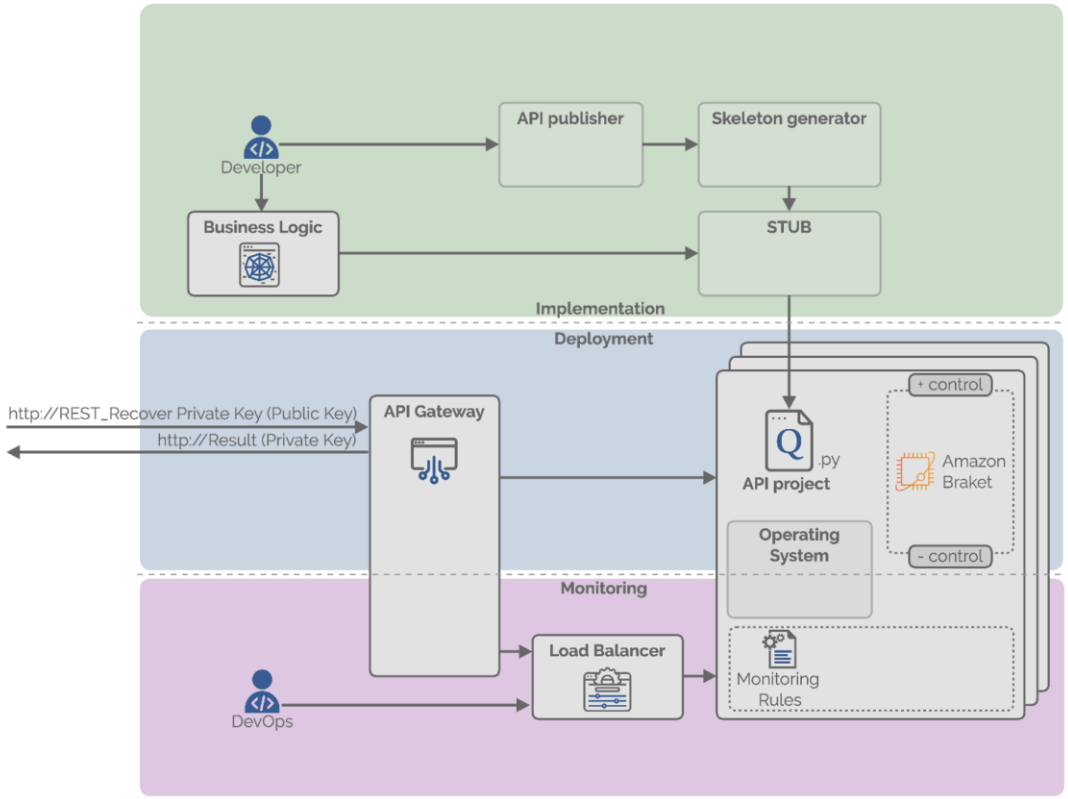
Quantum Web Engineering



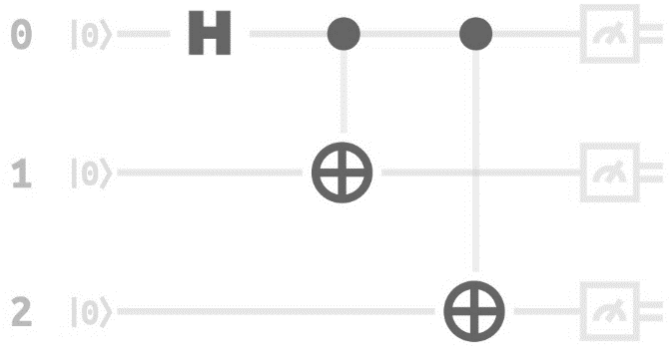
Quantum Web Engineering

There is no support for real services so X-Abilities are lost

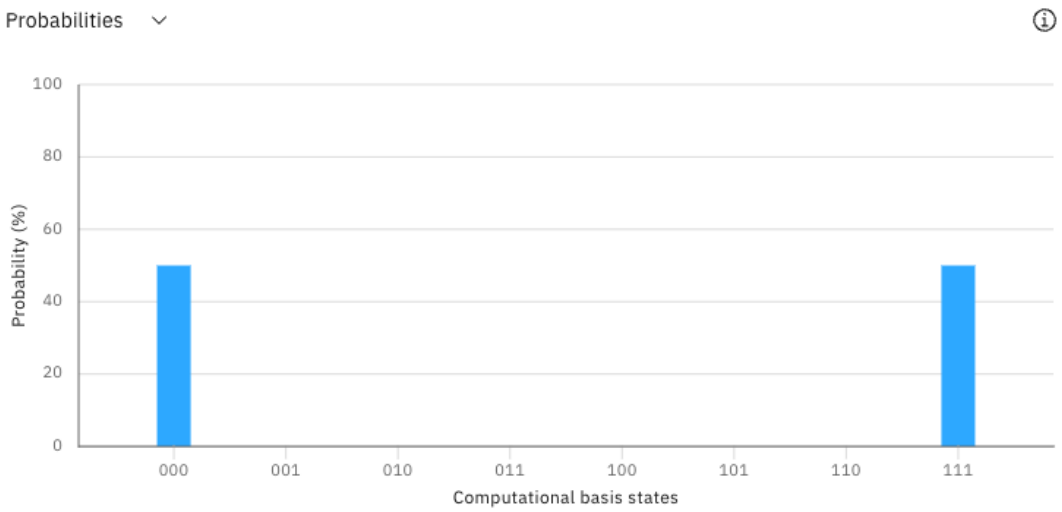




Quantum Web Eng – Current possibilities



We have **Quantum Code** providing a functionality and we would like to integrate it in our **app**



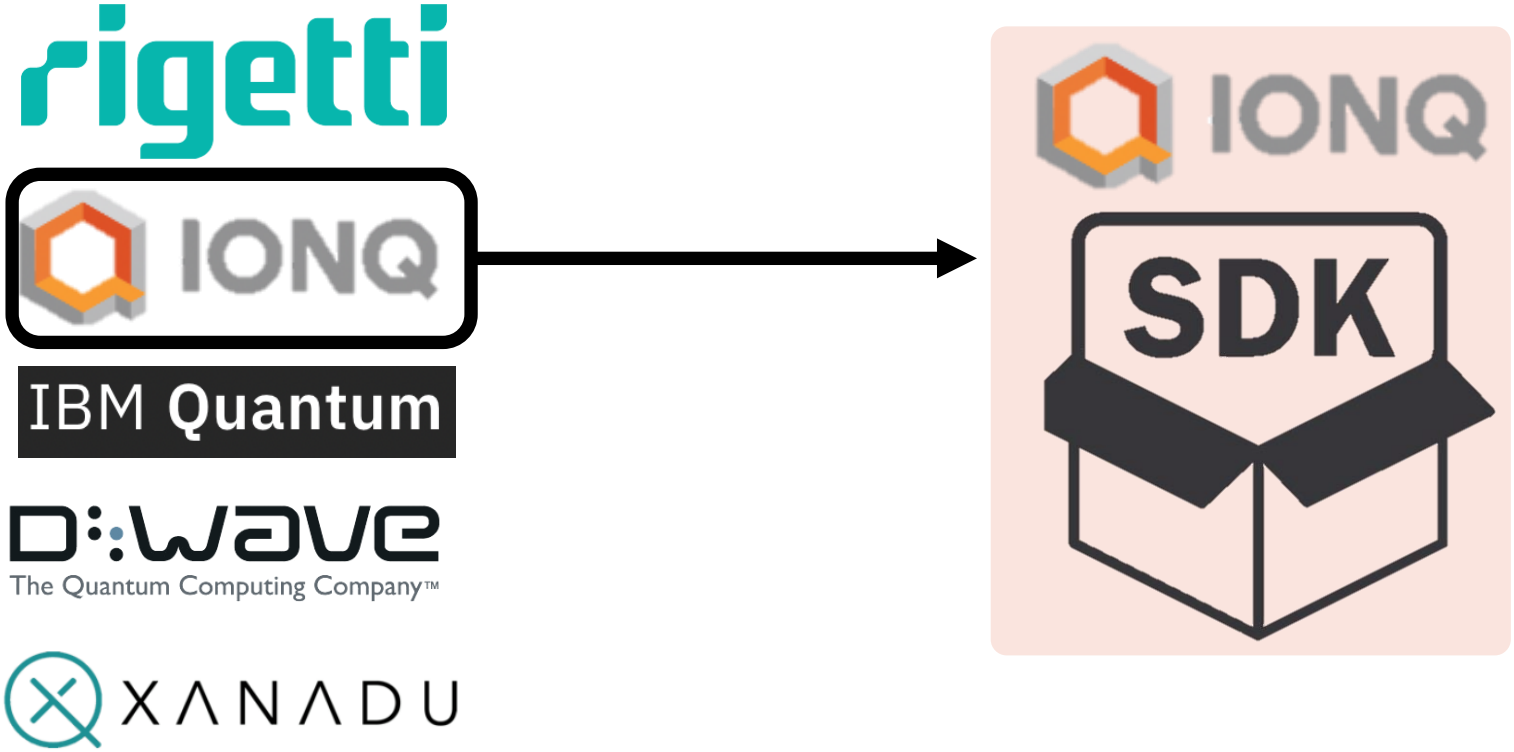
CLIENT APP

{Quantum Service Call}

Quantum Web Eng – Current possibilities

1. Use a hardware provider

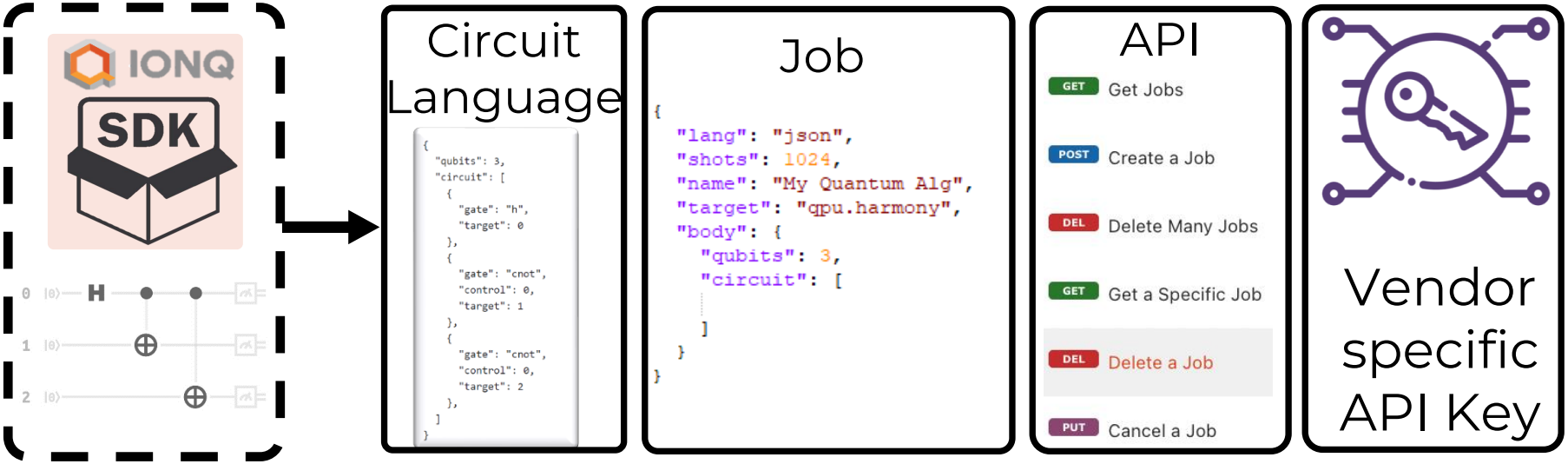
   Step 1: choose the provider



Quantum Web Eng – Current possibilities

1. Use a hardware provider

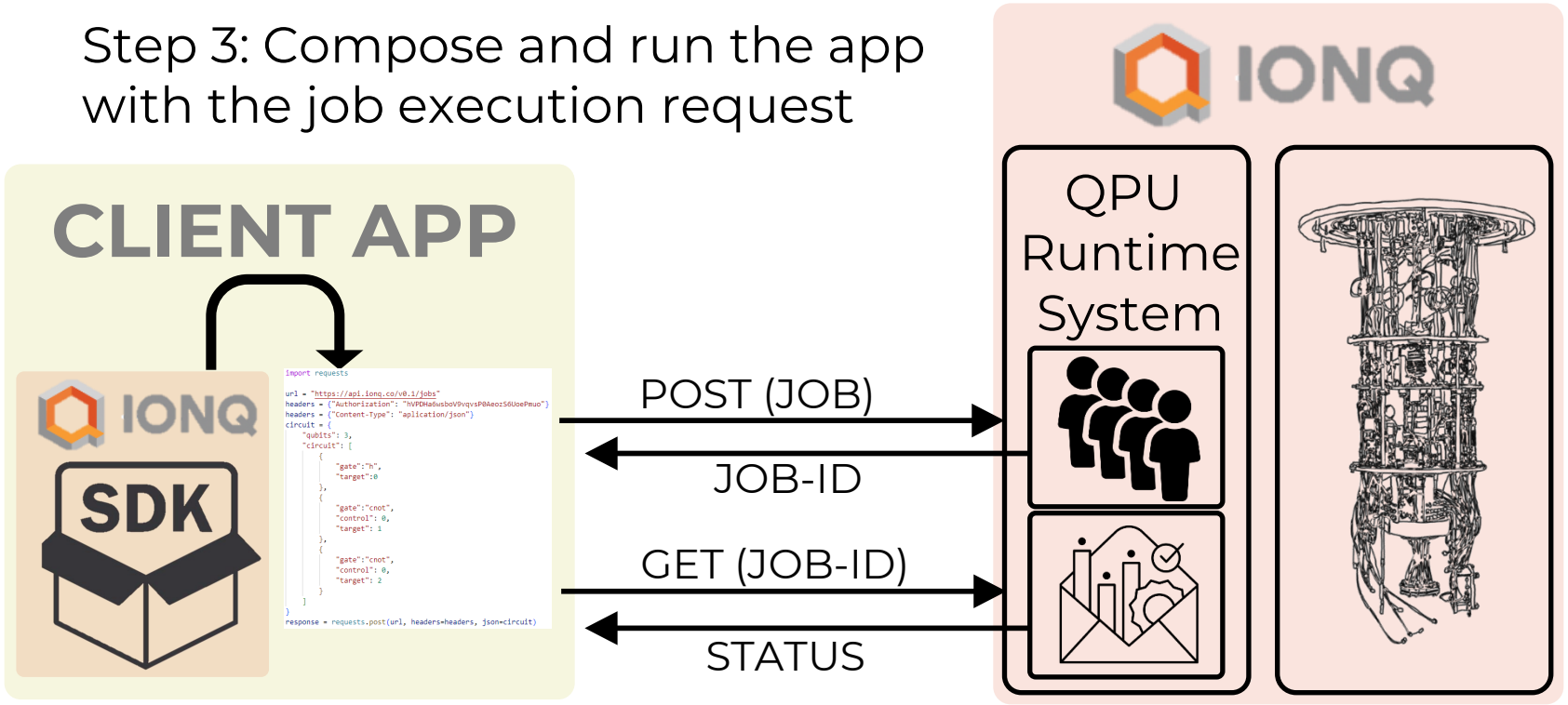
   Step 2: Use the provider's SDK to develop the quantum program and to integrate it in our app



Quantum Web Eng – Current possibilities

1. Use a hardware provider

Step 3: Compose and run the app with the job execution request



Quantum Web Eng – Current possibilities

1. Use a hardware provider

Step 3: Compose and run the app with the job execution request


```
import requests
url = "https://api.ionq.co/v0.1/jobs"
headers = {"Authorization": "hVPDHa6wsboV9vqvsP0AeozS6UoePmuo"}
headers = {"Content-Type": "aplication/json"}
circuit = {
    "qubits": 3,
    "circuit": [
        {
            "gate":"h",
            "target":0
        },
        {
            "gate":"cnot",
            "control": 0,
            "target": 1
        },
        {
            "gate":"cnot",
            "control": 0,
            "target": 2
        }
    ]
}
response = requests.post(url, headers=headers, json=circuit)
```

POST (JOB)

JOB-ID

GET (JOB-ID)

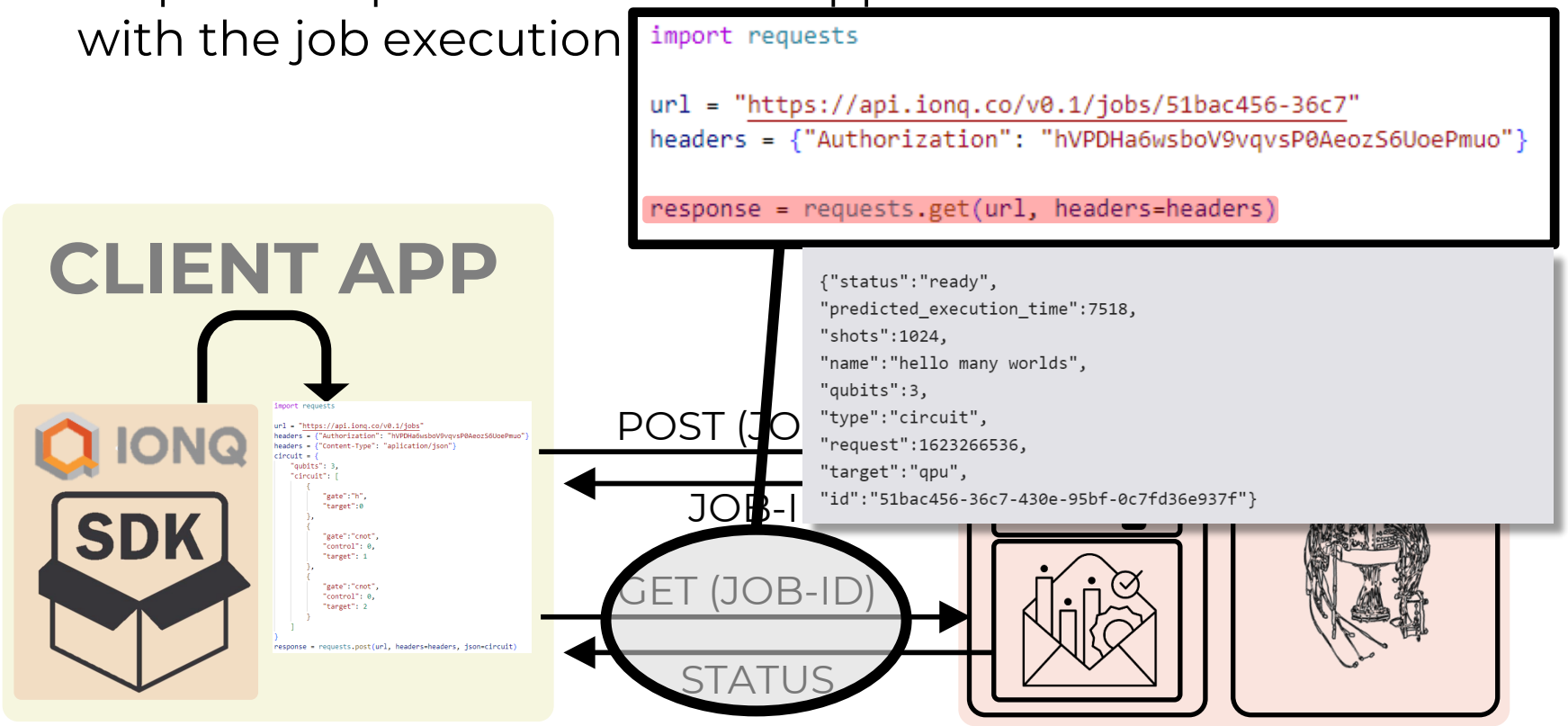
STATUS

```
{"id":"51bac456-36c7-430e-95bf-0c7fd36e937f",
"status":"ready",
"request":1623266536}
```

Quantum Web Eng – Current possibilities

1. Use a hardware provider

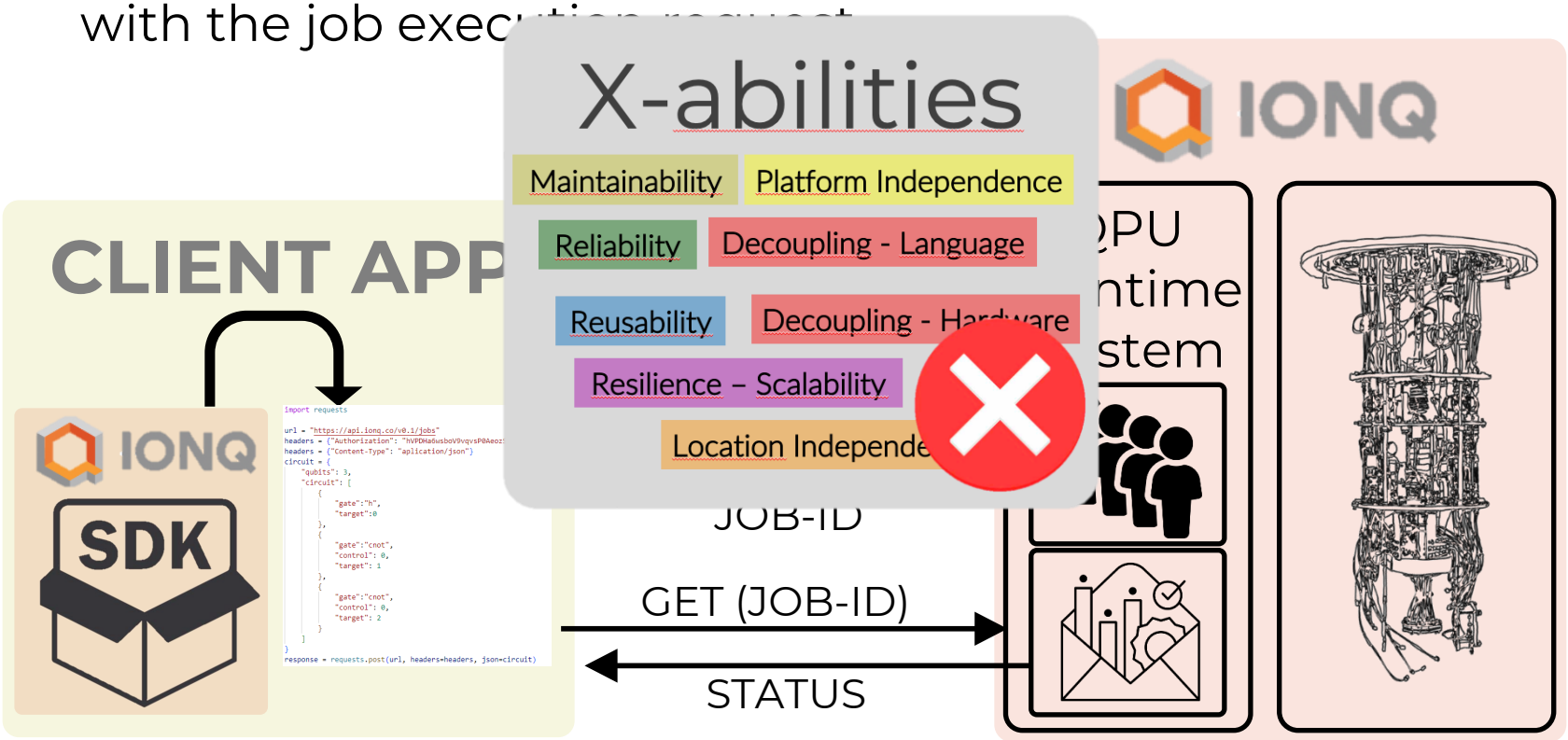
Step 3: Compose and run the app with the job execution



Quantum Web Eng – Current possibilities

1. Use a hardware provider

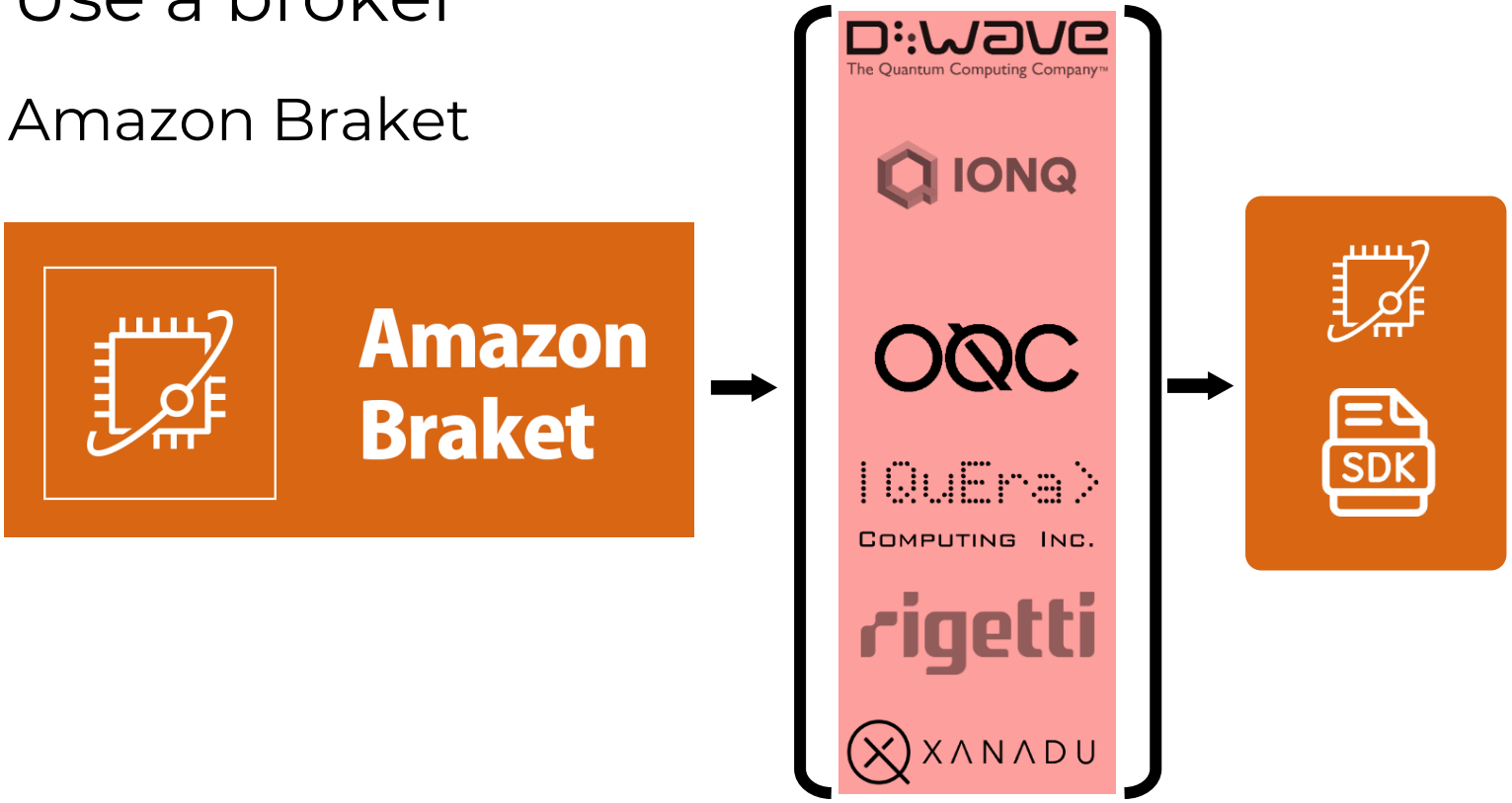
Step 3: Compose and run the app with the job execution request



Quantum Web Eng – Current possibilities

2. Use a broker

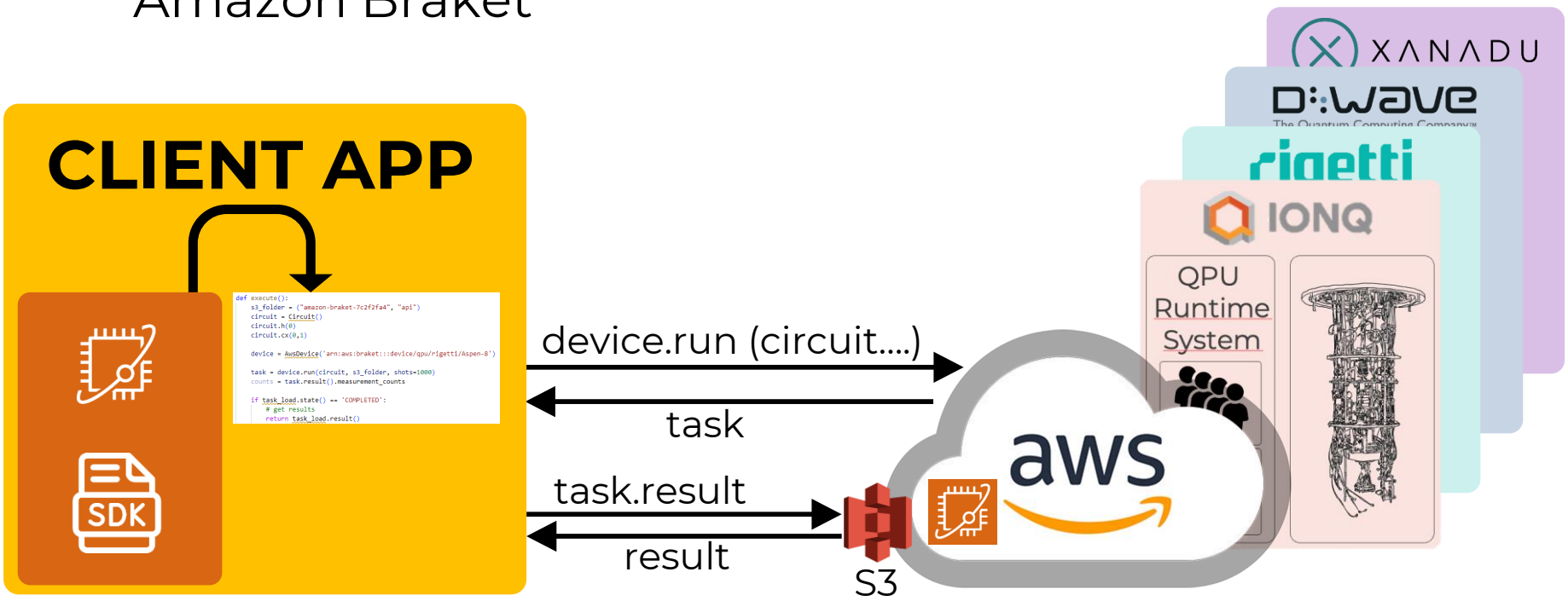
Amazon Braket



Quantum Web Eng – Current possibilities

2. Use a broker

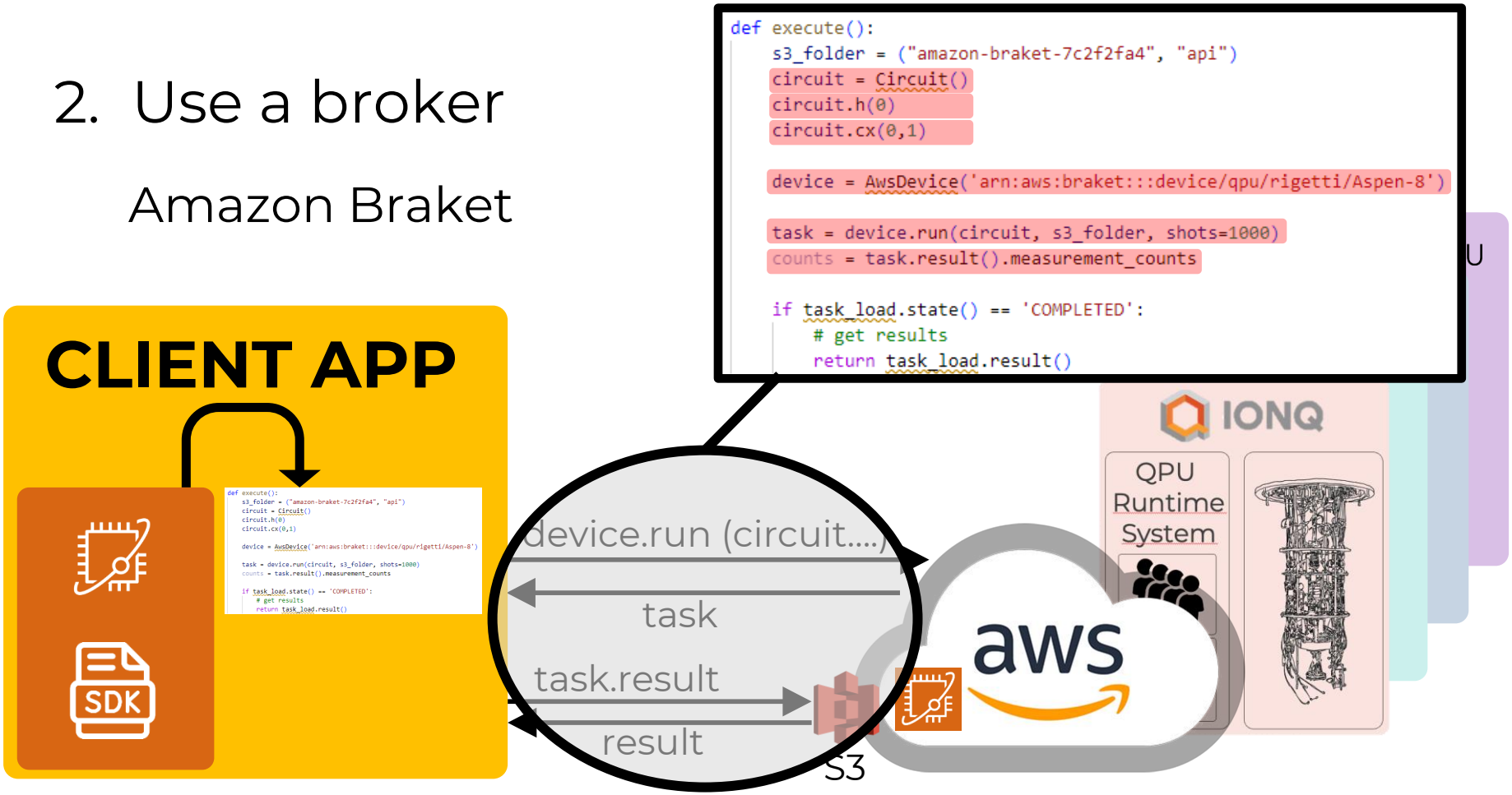
   Amazon Braket



Quantum Web Eng – Current possibilities

2. Use a broker

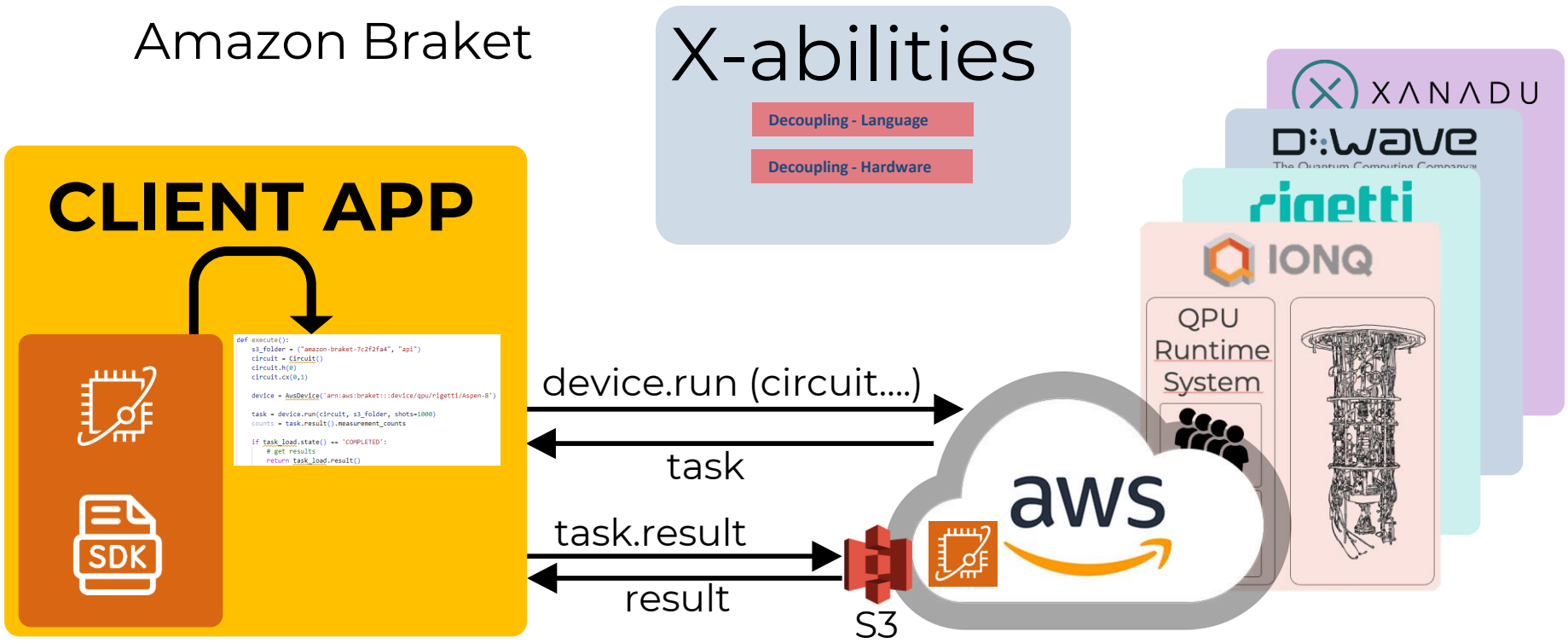
   Amazon Braket



Quantum Web Eng – Current possibilities

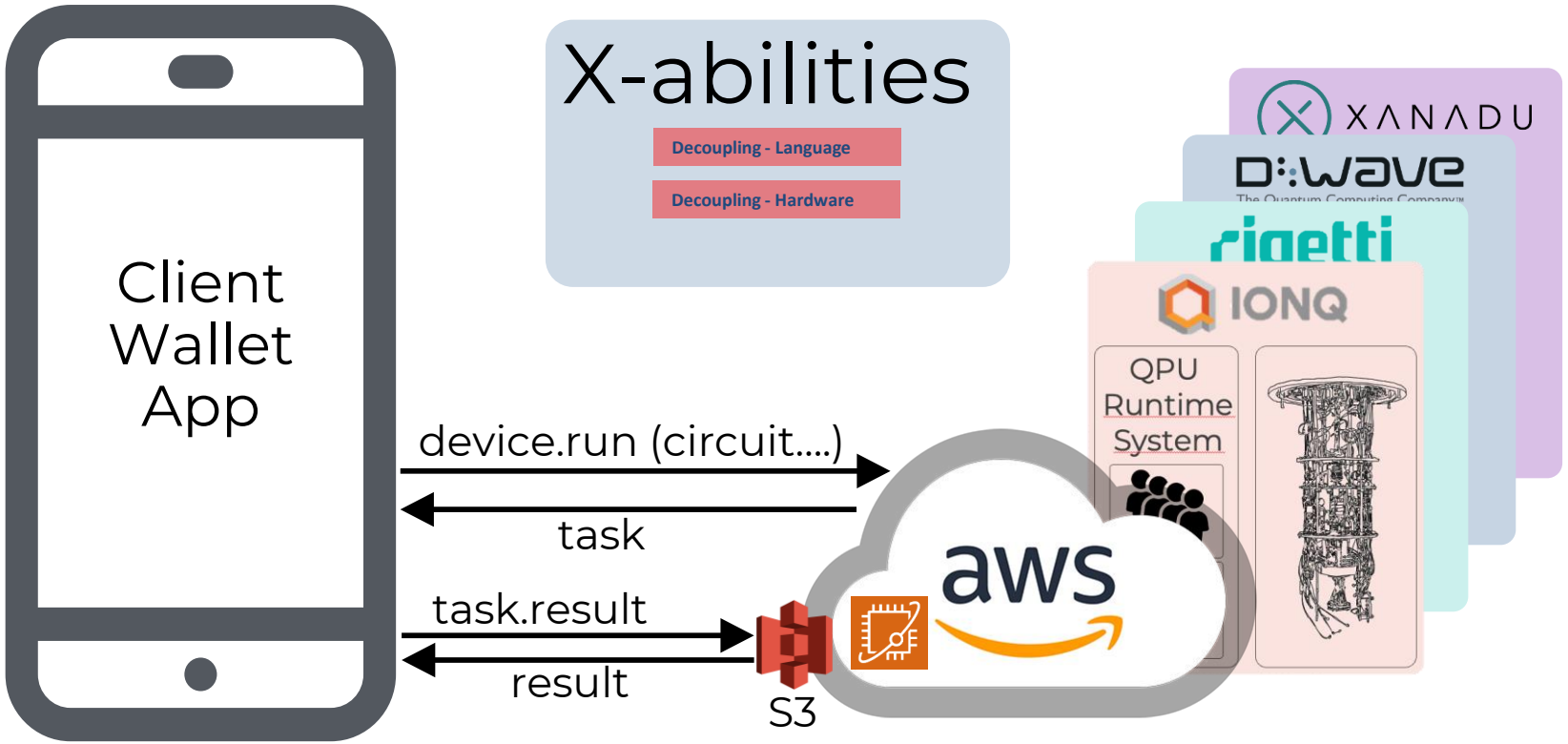
2. Use a broker

Amazon Braket



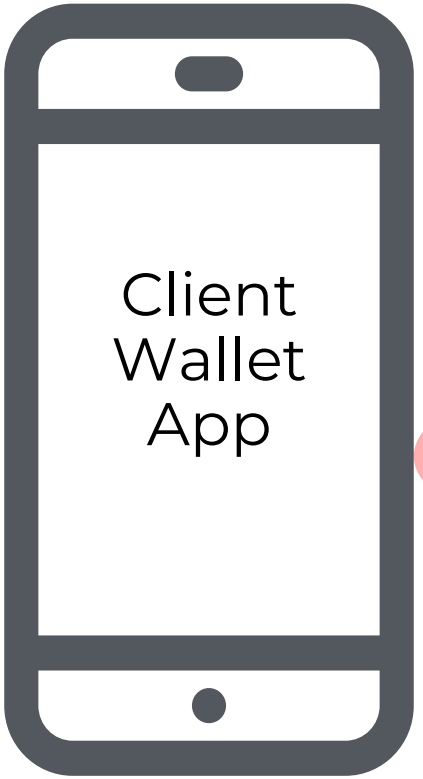
Quantum Web Eng – Current possibilities

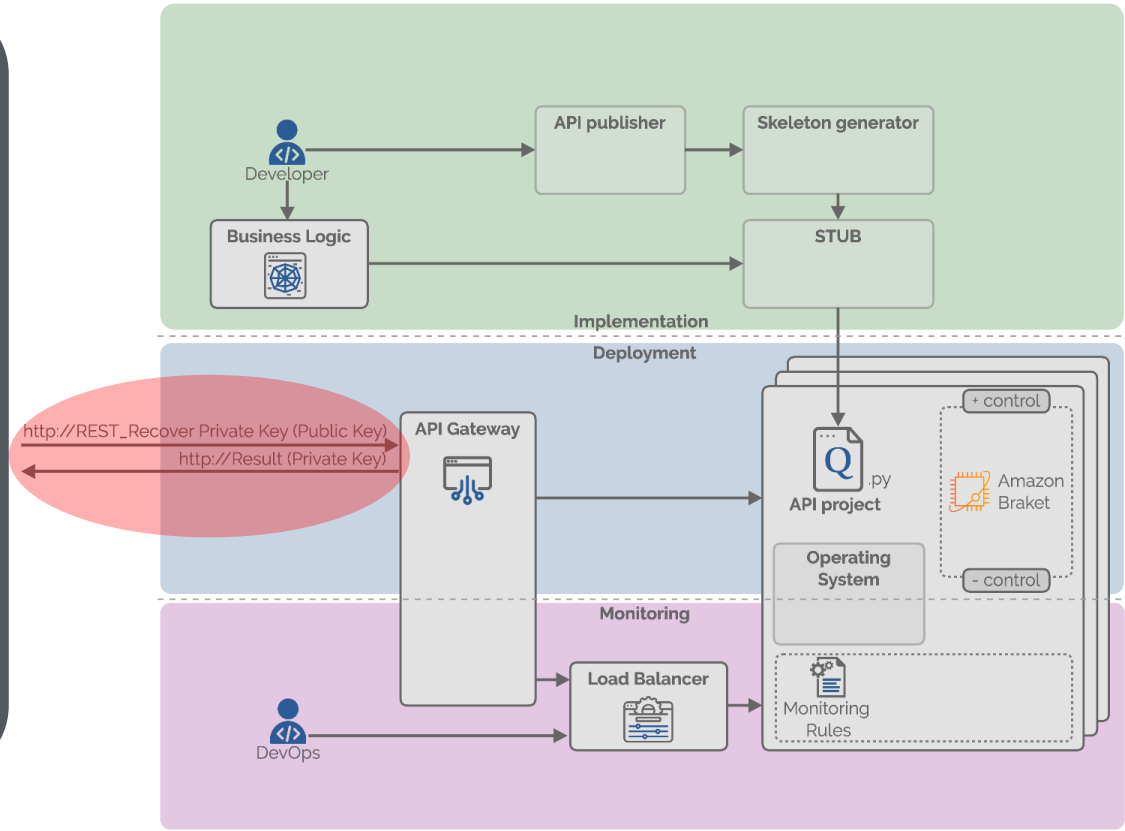
2. Use a broker



Servitizing quantum circuits

Objective 1
To have service calls instead of program execution requests





Servitizing quantum circuits

We started by encapsulating a quantum circuit inside a classical service acting as a wrapper.

```python
from flask import Flask, request, jsonify, send_file
from flask_cors import CORS
import matplotlib.pyplot as plt

from braket.circuits import Circuit
from braket.devices import LocalSimulator

app = Flask(__name__)
CORS(app)

@app.route('/execute', methods=["get"])
def execute_quantum_task():
    bell = Circuit().h(0).cnot(control=0, target=1)
    device = LocalSimulator()
    result = device.run(bell, shots=1000).result()
    counts = result.measurement_counts

    plt.bar(counts.keys(), counts.values())
    plt.xlabel('bitstrings')
    plt.ylabel('counts')
    plt.savefig("result.png")

    return send_file("result.png", mimetype='image/png')

if __name__ == '__main__':
    app.run(host="localhost", port=33888)
```

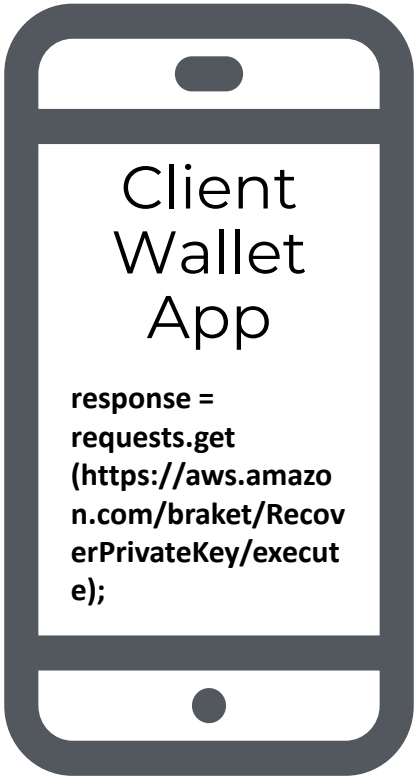
Braket libraries for quantum computing

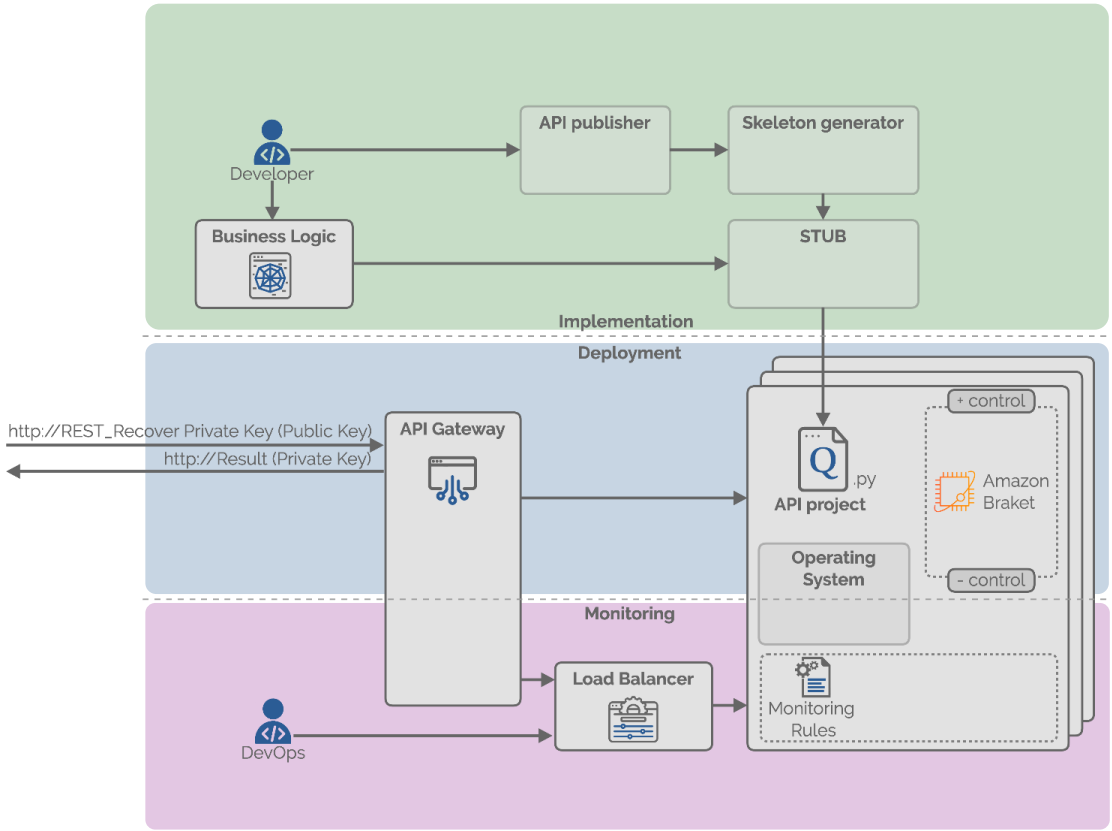
Classical wrapping service

Quantum algorithm

Servitizing quantum circuits

Objective 1
To have service calls instead of program execution requests



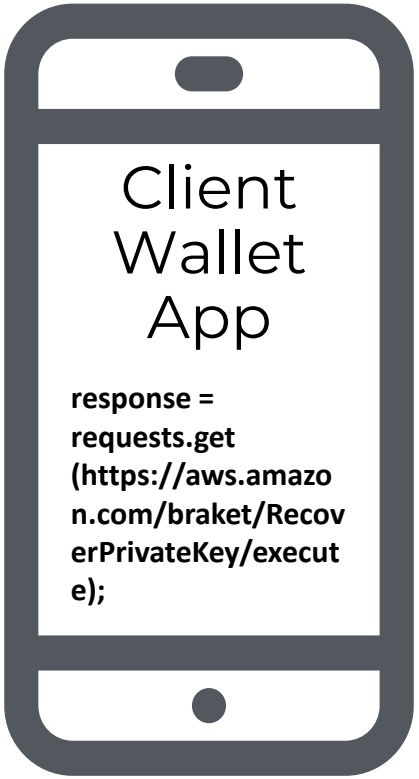


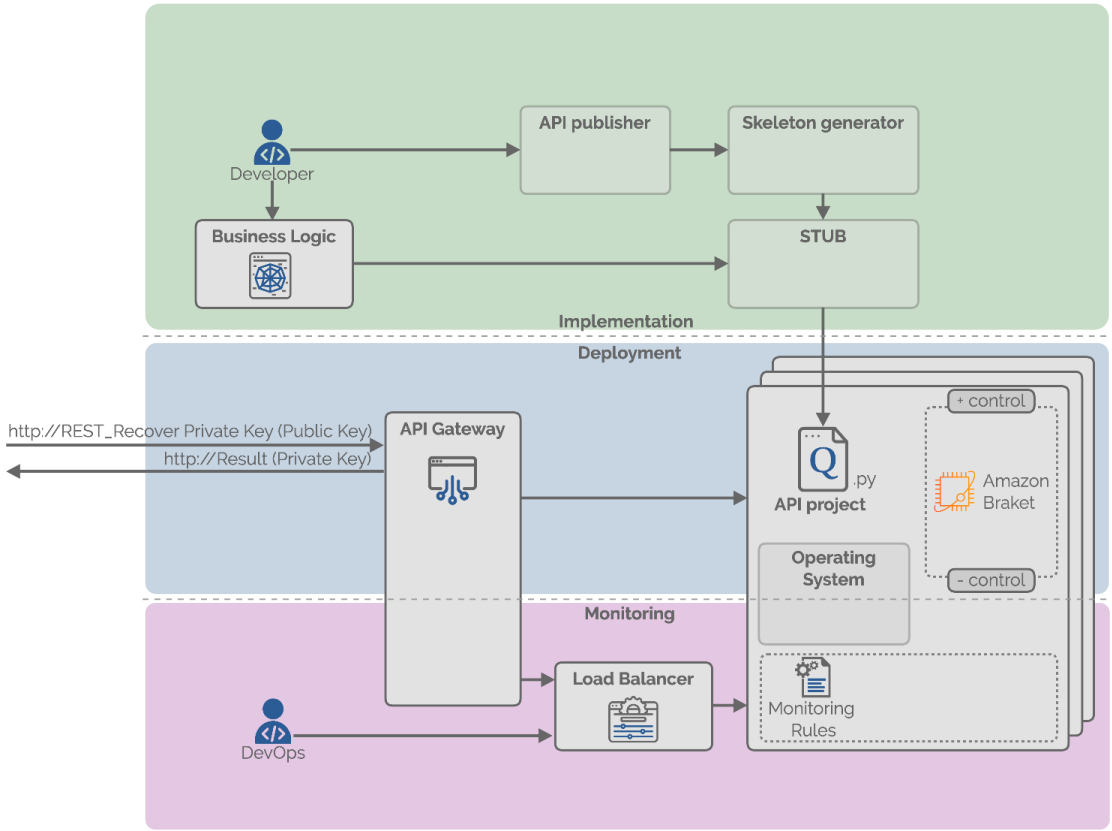
Servitizing quantum circuits

Objective 1

X-abilities

- Decoupling - Language
- Decoupling - Hardware
- Location Independence
- Manteinability
- Reusability

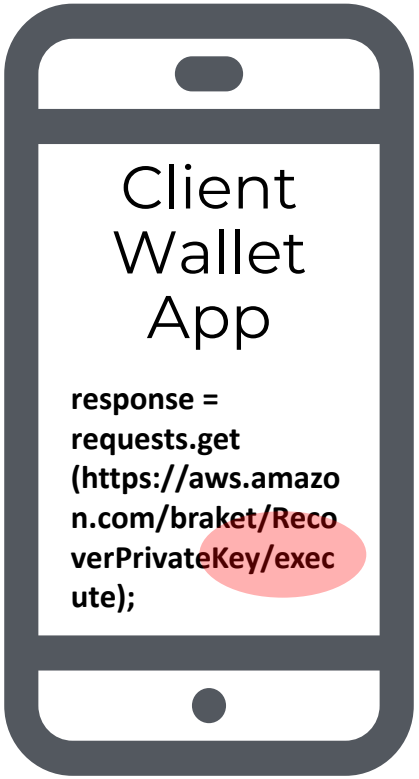


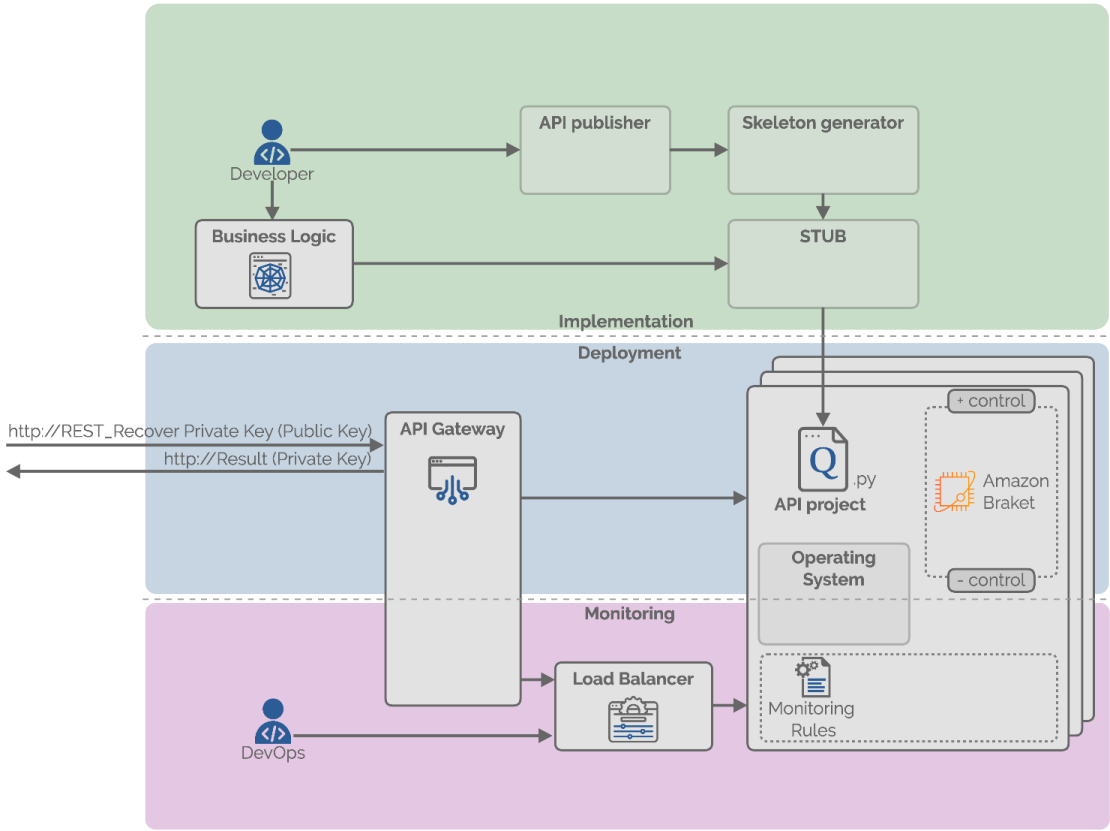


Quantum SOC – Servitizing quantum circuits

Objective 2

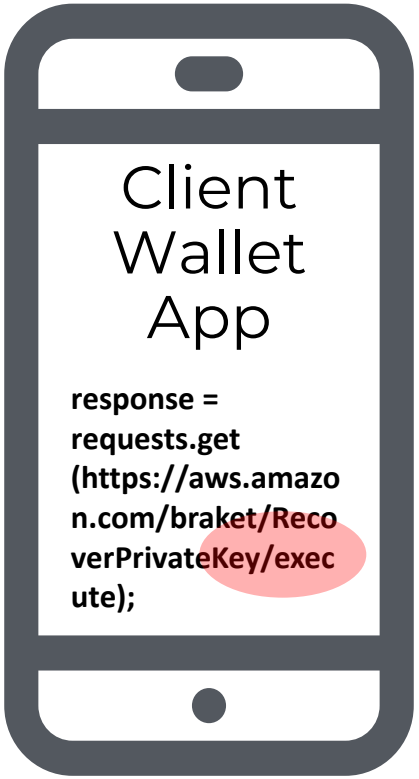
To parameterize the service call





Servitizing quantum circuits

Objective 2
To parameterize the service call



What kind of parameters?

1. Regular parameters for services (eg. Public_key to regenerate private_key)

2. Specific parameters for the quantum execution
   a. Number of shoots
   b. Time (maximum)
   c. Cost (maximum)
   d. The QCaaS provider

Servitizing quantum circuits

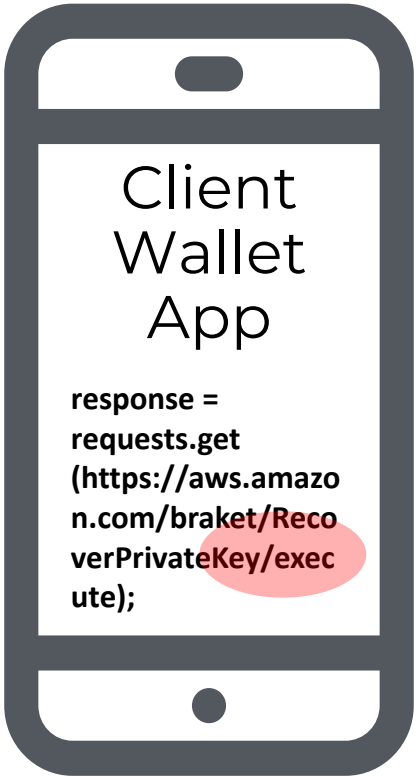
The set of available QCaaS providers is continuously growing

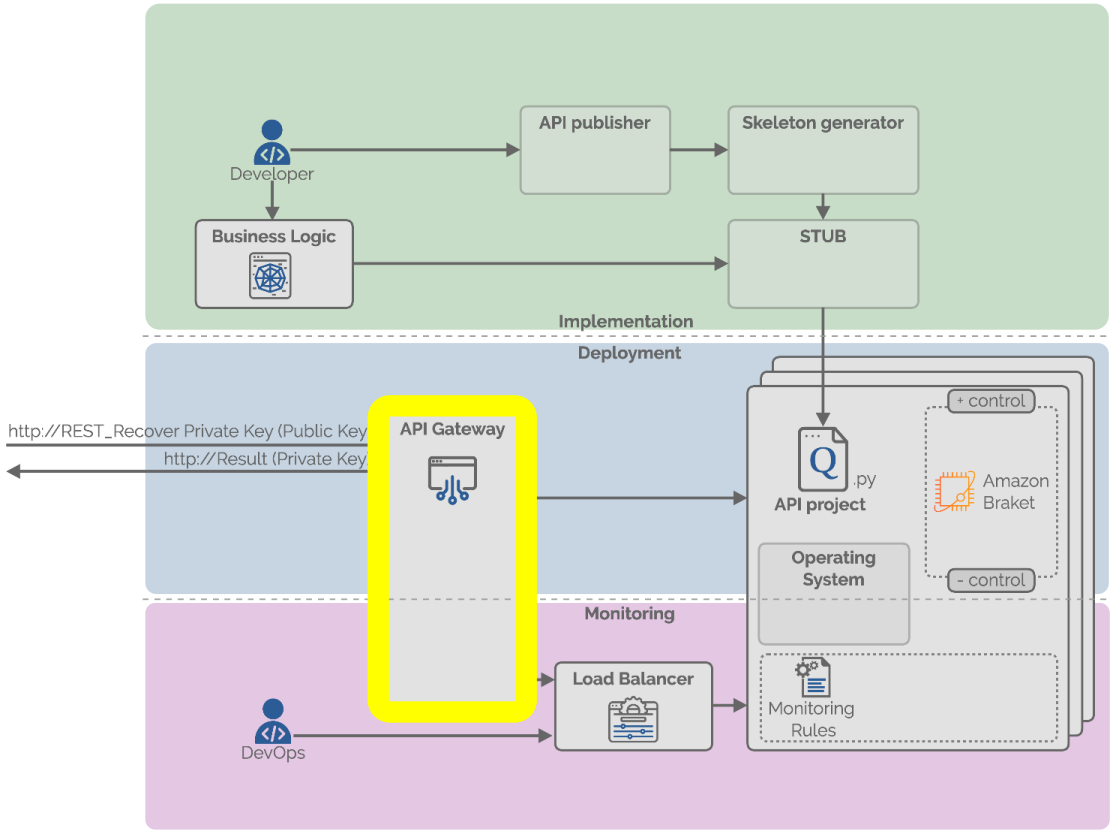
Libraries	Languages	Cloud Applications	Quantum Computers	Quantum Simulators	Quantum Providers
Qiskit OpenQASM	Python	IBM Cloud	• IonQ computer • IBM computers (+20)	• Qiskit Aer • simulator_stabilizer • simulator_statevector • simulator_mps • ibmq_qasm_simulator	IBM Q IBM Quantum
Cirq PennyLane	Python	Google Cloud	• Google Sycamore	• Qsim • Qsimh • Stim	Google Quantum AI
Q# libs Cirq Qiskit	Q# Python	Microsoft Azure	• IonQ computer • Toshiba SBM • QCI machines • Honeywell quantum • Pasqal computers	• QDK Simulator	Azure Quantum
Amazon Braket	Python	amazon web services	• Rigetti computers (Aspen-11, Aspen M-2) • Xanadu Borealis • IoQ computer • braket_sv • braket_dm • OQC • D-Wave computers (*quantum annealer*)	• Local state vector (braket_sv) • State vector (SV1) • Density matrix simulator (DM1) • Tensor network simulator (TN1) • PennyLane's Lightning Simulators	Amazon Braket

Servitizing quantum circuits

Objective 2

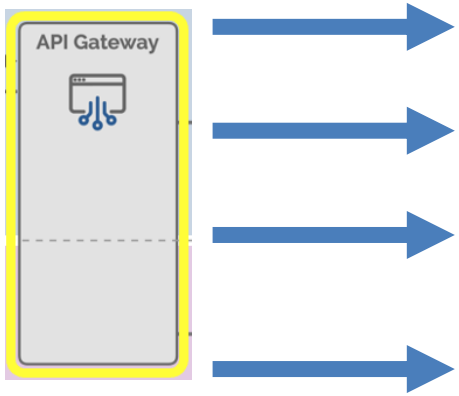
To parameterize the service call





Servitizing quantum circuits

Empower the API Gateway providing it with Quantum awareness: the Quantum API Gateway



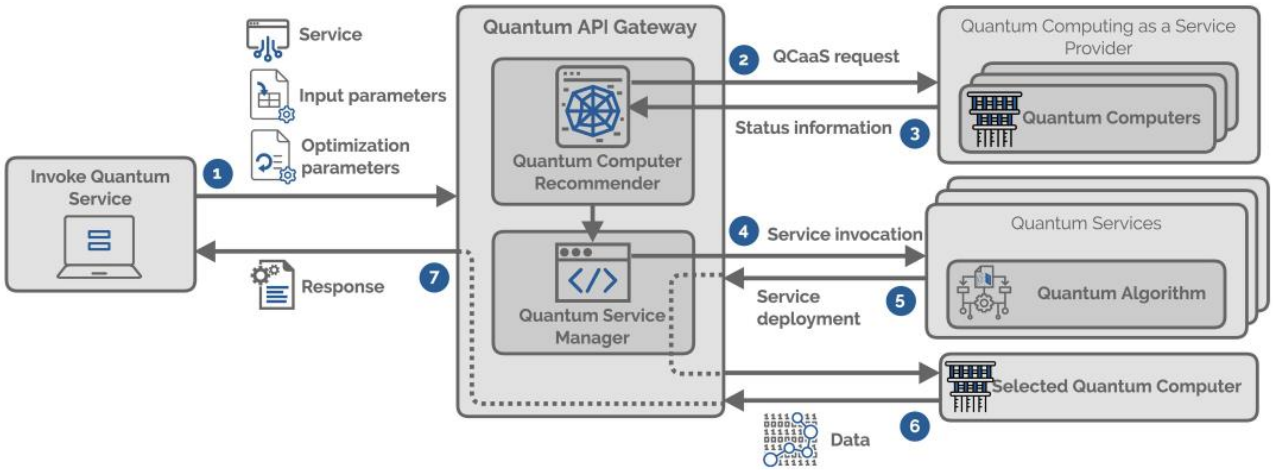
Libraries	Languages	Cloud Applications	Quantum Computers	Quantum Simulators	Quantum Providers
Qiskit OpenQASM	Python	IBM Cloud	• IonQ computer • IBM computers (+20)	• Qiskit Aer • simulator_stabilizer • simulator_statevector • simulator_mps • ibmq_qasm_simulator	IBM Q IBM Quantum
Cirq PennyLane	Python	Google Cloud	• Google Sycamore	• Qsim • Qsimh • Stim	Google Quantum AI
Q# libs Cirq Qiskit	Q# Python	Microsoft Azure	• IonQ computer • Toshiba SBM • QCI machines • Honeywell quantum • Pasqal computers	• QDK Simulator	Azure Quantum
Amazon Braket	Python	amazon web services	• Rigetti computers (Aspen-11, Aspen M-2) • Xanadu Borealis • IoQ computer • braket_sv • braket_dm • OQC • D-Wave computers (*quantum annealer*)	• Local state vector (braket_sv) • State vector (SV1) • Density matrix simulator (DM1) • Tensor network simulator (TN1) • PennyLane's Lightning Simulators	Amazon Braket

Servitizing quantum circuits

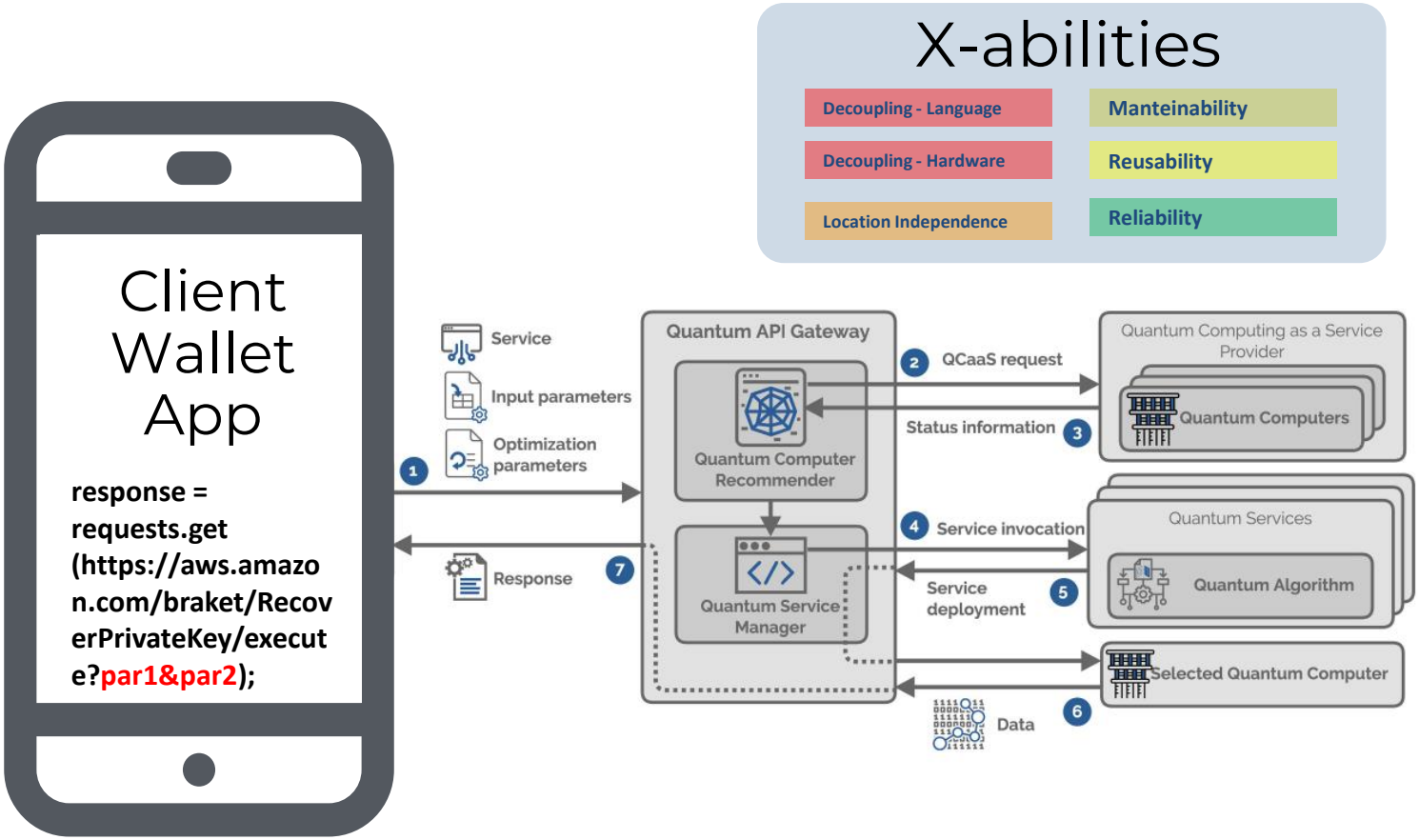
THEME ARTICLE: QUANTUM AND POST-MOORE'S LAW COMPUTING

Quantum Software as a Service Through a Quantum API Gateway

Jose Garcia-Alonso, Javier Rojo, David Valencia, Enrique Moguel, Javier Berrocal, and Juan Manuel Murillo, *University of Extremadura, 10003 Cáceres, Spain*



Servitizing quantum circuits

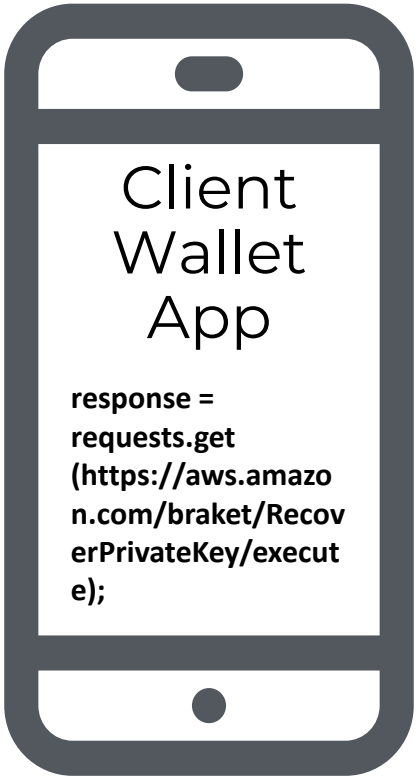


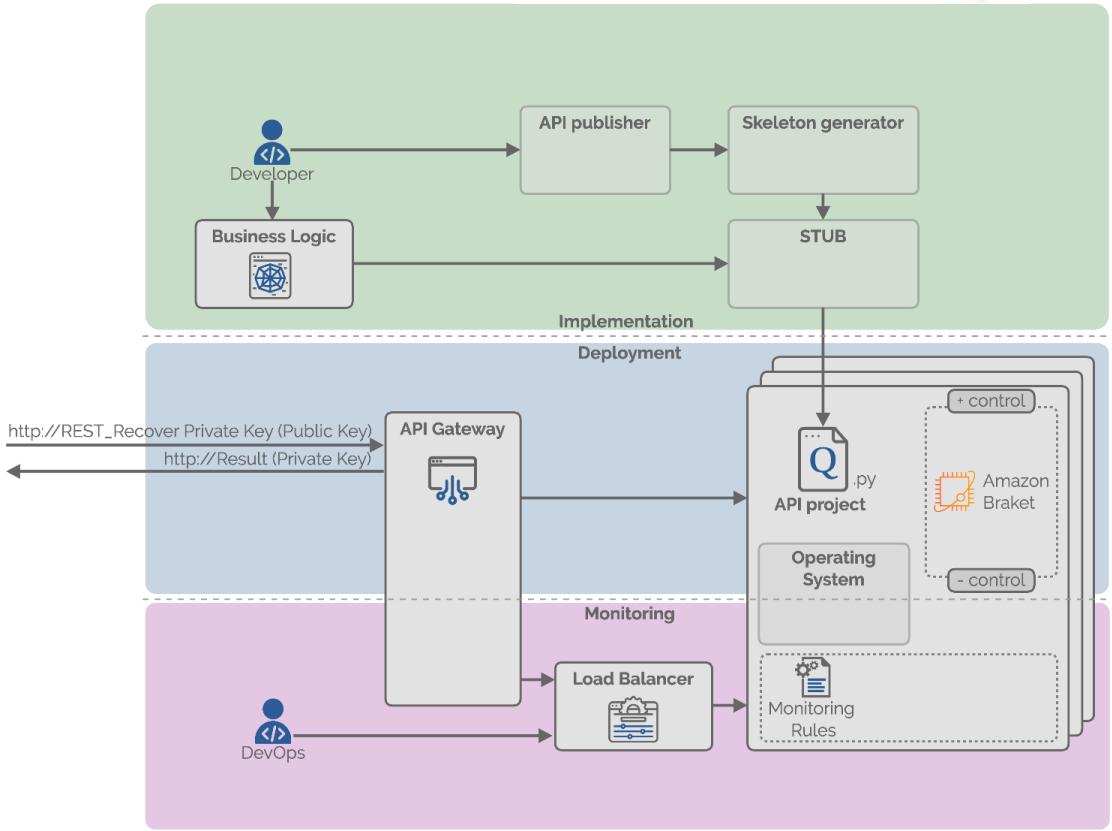
Servitizing quantum circuits

COMING SOON

Objective 3

Provide developers with tools like the ones they typically use to build and deploy services





Conclusions

- QC and Web Engineering
  - QC is already a trend in Web Engineering and Service-Oriented Computing research
  - Industry is following close
    - Quantum workforce needed
    - German Quantum Initiative…
  - No need to be a Quantum Physicist
    - Web engineers can already provide significant contributions to Quantum Computing

Danke schön!

SPILab by Quercus SEG

TECHNISCHE UNIVERSITÄT CHEMNITZ