IEEE QUANTUM WEEK

QIC

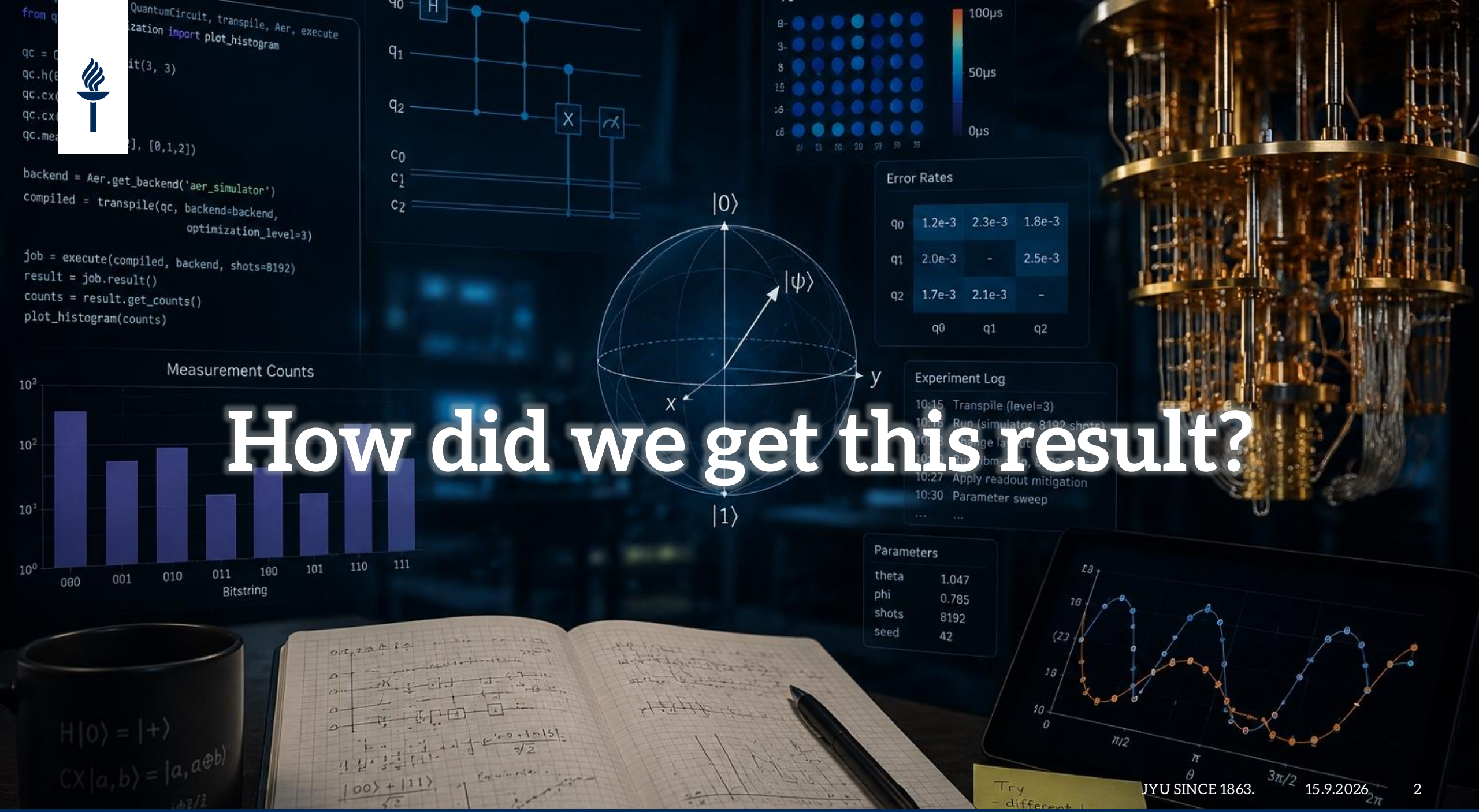
JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

From Circuits to Results
Systematic Experiment Tracking in Quantum Workflows

Vlad Stirbu

13.09.2026

Toronto, Canada



How did we get this result?

Vlad Stirbu

Otso Kinanen

Valter Uotila

Oskari Kerppo

Foundations

What is experiment tracking

Theory and purpose

Why we should care

Experiment tracking for quantum

What to track: circuit, hardware, compilation, execution

Design experiment

Setting up the environment

Mlflow, User interface

How to track

Applied practice

Quantum Reservoir Computing

Motivation and setitng the experiment

Stages: preprocessing, execution, error mitigation

Benefits

Outlook

Quantum CFD

Intro to CFD problems

Tool for multi-stage algorithms

Benefits

What a quantum experiment looks like in the paper

What producing the result actually looks like

- Which circuit did they actually execute?
- Which parameters / input data?
- Which compiler/transpiler configuration and passes transformed it?
- Which SDK and package versions?
- Which backend: simulator or QPU?
- How many shots?
- What was the hardware configuration/calibration state?
- Which mitigation/post-processing steps?
- Which intermediate results led to choose this experiment?
- Can somebody else reconstruct it (six months later)?

My very first quantum program 🤩

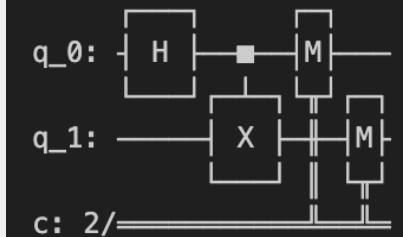
1
```
pip install qiskit
```

Circuit → Execution → Result

2
```
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator

circuit = QuantumCircuit(2)
circuit.h(0)
circuit.cx(0, 1)

simulator = AerSimulator()
result = simulator.run(circuit).result()
counts = result.get_counts()

print("Total count for 00 and 11 are:", counts)
print(circuit)
```

3
```
python app.py
```

4
```
Total count for 00 and 11 are: {'11': 1007, '00': 993}
```



Circuit preparation

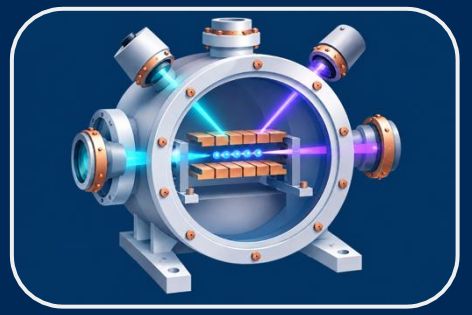
```python
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator

circuit = QuantumCircuit(2, 2)
circuit.h(0)
circuit.cx(0, 1)
circuit.measure([0, 1], [0, 1])

simulator = AerSimulator()
result = simulator.run(circuit).result()
counts = result.get_counts()
```

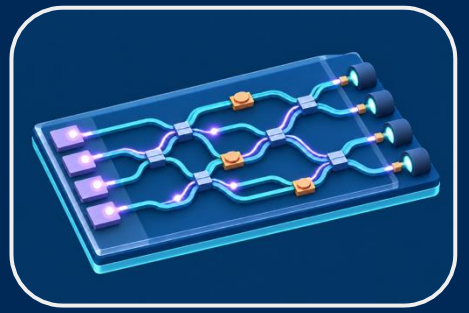
```python
import qiskit


# My s3cret program
```

```python
from mqt.bench import BenchmarkLevel, get_benchmark


circuit = get_benchmark(
  benchmark="qft",
  level=BenchmarkLevel.ALG,
  circuit_size=30,
)
```

MQSS
Munich Quantum Software Stack

Execution on hardware





```python
from qiskit import QuantumCircuit, transpile
from iqm.qiskit_iqm import IQMProvider

circuit = QuantumCircuit(2, 2)
circuit.h(0)
circuit.cx(0, 1)
circuit.measure([0, 1], [0, 1])

backend = IQMProvider("https://qx.vtt.fi/api/devices/demo")
backend = backend.get_backend()

transpiled = transpile(circuit, backend)

result = backend.run(transpiled, shots=1000).result()
```

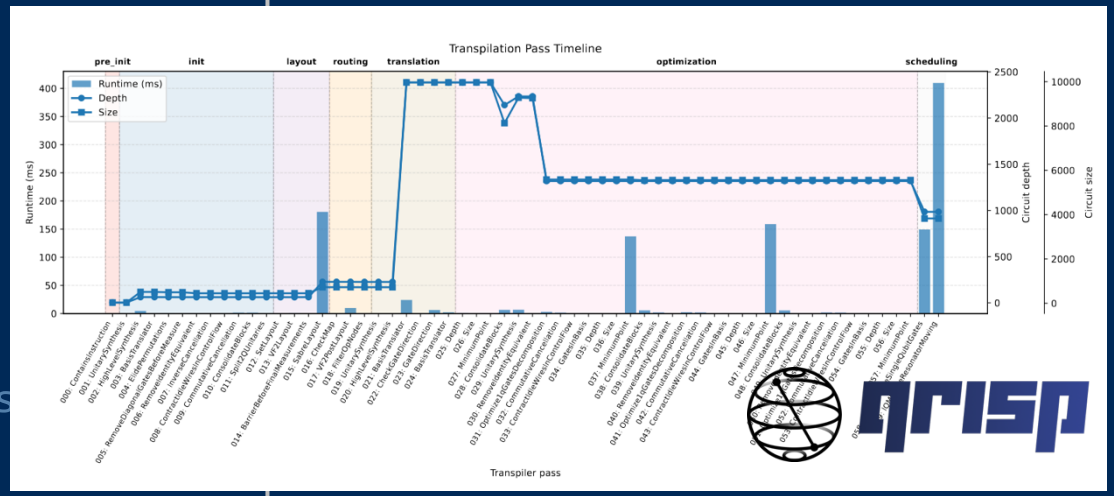
Provider	Circuit	Computer	Compilation	Execution	Software
Hardware Providers					
AQT					
D-Wave					
Google Quantum AI					
IBM					
IonQ					
IQM					
Pasqal					
Quandela					
Quantinuum					
QuEra					
Rigetti					
VTT					
Cloud Aggregators					
Amazon Braket					
Microsoft Azure Quantum					
qBraid					

Legend: comprehensive (broad coverage of applicable attributes); partial (substantial coverage with notable gaps); limited (few or no applicable attributes exposed)

https://arxiv.org/abs/2608.08272

Compiler observability

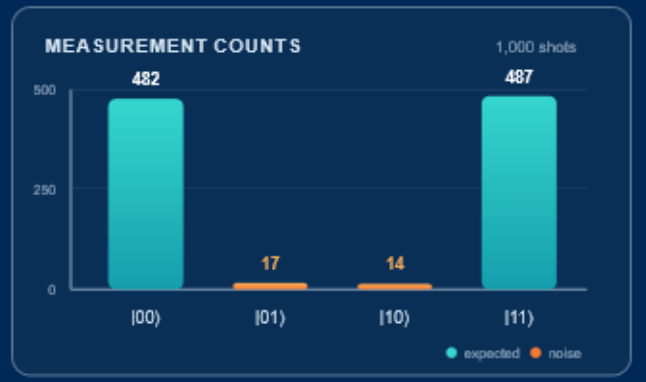
```python
from qiskit import QuantumCircuit, transpile
from iqm.qiskit_iqm import IQMProvider

circuit = QuantumCircuit(2, 2)
circuit.h(0)
circuit.cx(0, 1)
circuit.measure([0, 1], [0, 1])

backend = IQMProvider("https://qx.vtt.fi/api/devices
backend = backend.get_backend()

transpiled = transpile(circuit, backend)

result = backend.run(transpiled, shots=1000).result()
```



https://arxiv.org/abs/2608.05041

Process the results

```python
from qiskit import QuantumCircuit, transpile
from iqm.qiskit_iqm import IQMProvider

circuit = QuantumCircuit(2, 2)
circuit.h(0)
circuit.cx(0, 1)
circuit.measure([0, 1], [0, 1])

backend = IQMProvider("https://qx.vtt.fi/api/devices/demo")
backend = backend.get_backend()

transpiled = transpile(circuit, backend)

result = backend.run(transpiled, shots=1000).result()
```



post-process

RAW FIDELITY	MITIGATED FIDELITY
0.94 ± 0.02	0.98 ± 0.01

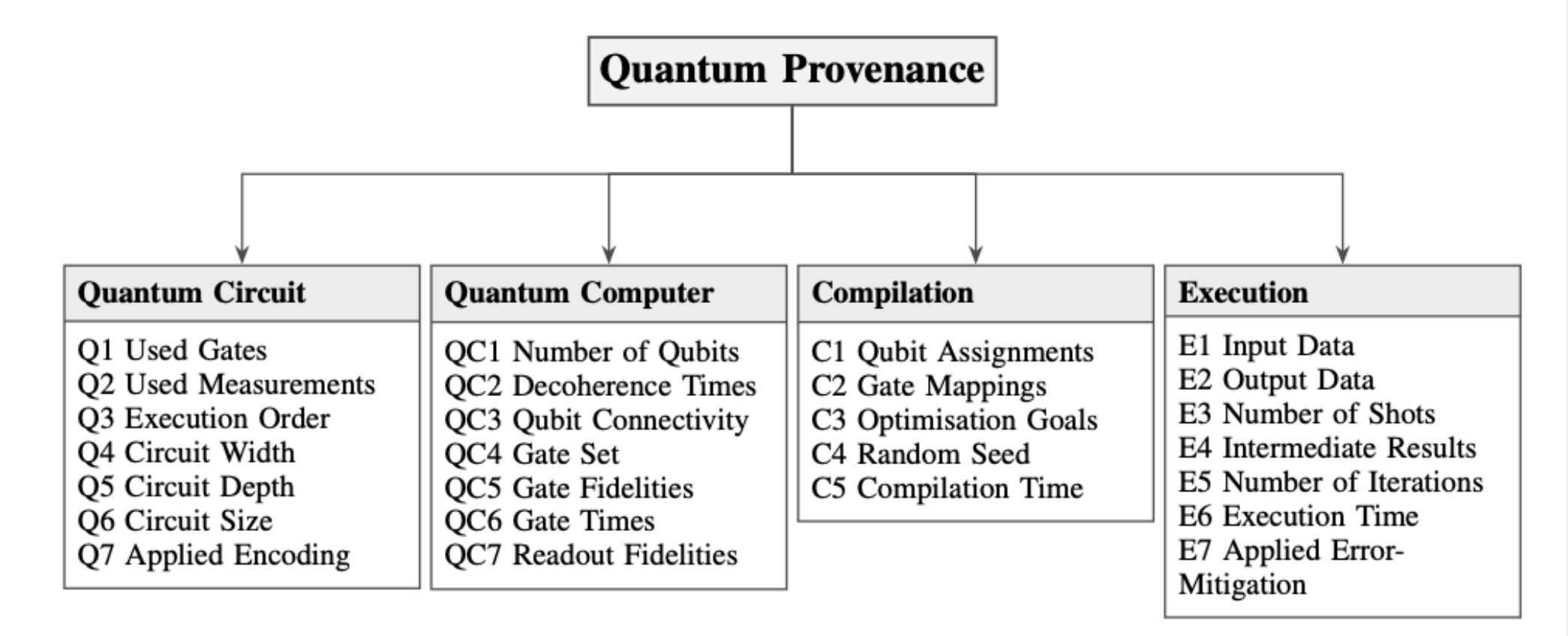
Software supply chain

```
from qiskit import QuantumCircuit, transpile
from iqm.qiskit_iqm import IQMProvider

circuit = QuantumCircuit(2, 2)
circuit.h(0)
circuit.cx(0, 1)
circuit.measure([0, 1], [0, 1])

backend = IQMProvider("https://qx.vtt.fi/api/devices/demo")
backend = backend.get_backend()

transpiled = transpile(circuit, backend)

result = backend.run(transpiled, shots=1000).result()
```

Six months later

- Can I **reproduce** this result?
- Can I **compare** it fairly with another result?
- Can I **explain** why it changed?

Quantum Provenance Schema



Weder, B., et al.: QProv: a provenance system for quantum computing, 2021

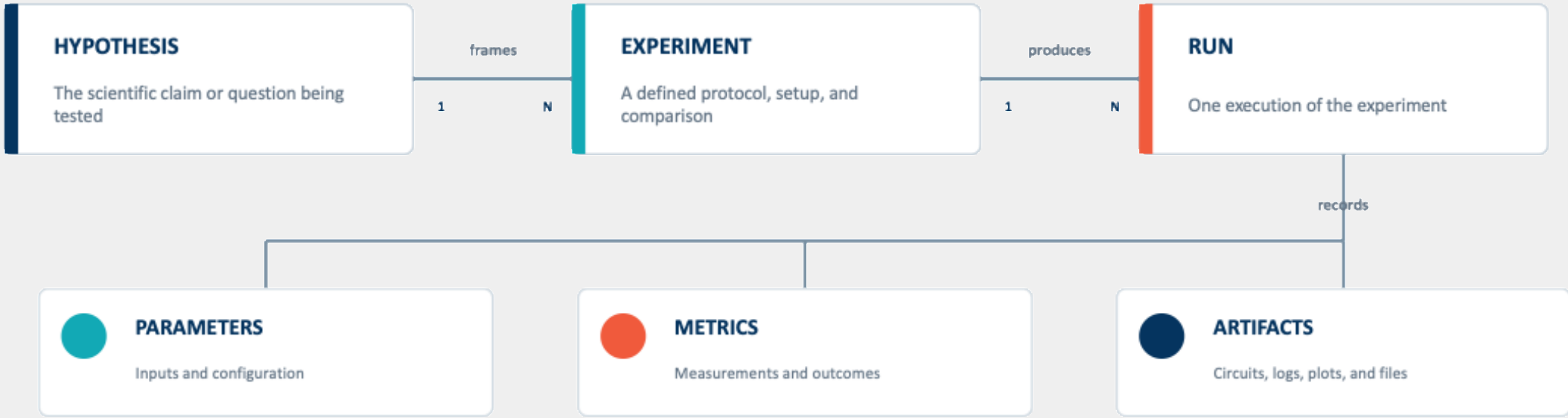
Experiment tracking

The systematic recording of the inputs, configuration, execution context, intermediate transformations, outputs, and artifacts associated with an experiment.

Engineering practice/Research logbook

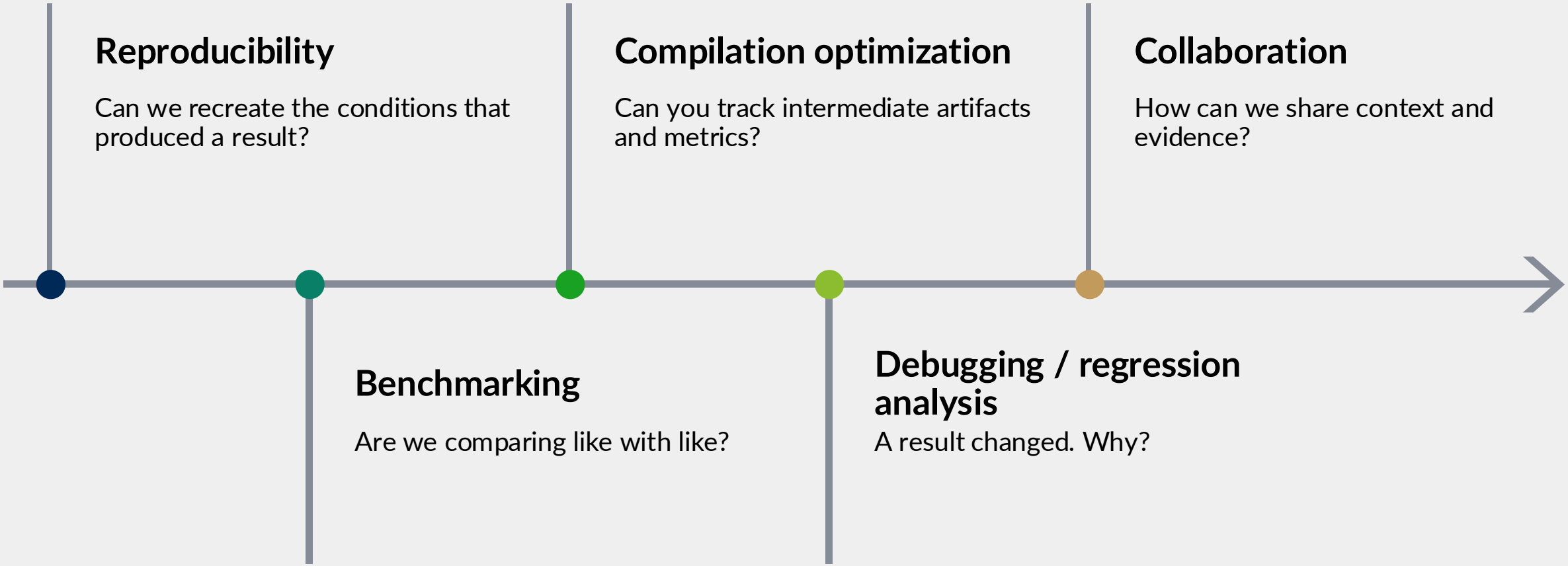
Experiment tracking in scientific context

A provenance model for connecting scientific intent to recorded evidence



Experiment tracking preserves these relationships so every result remains traceable to its scientific context.

Why track experiments?



Is experiment tracking only for developers?

Stakeholder	Concerns
Developer / researcher	Why did my result change? Can I reproduce it?
PI / lab manager	What experiments are we running and what evidence supports our conclusions?
Infrastructure provider	How are resources being used and how do workloads behave?
Hardware manufacturer	How does hardware behavior affect applications and workloads?

Developer

- reproducibility
- debugging
- benchmarking
- optimization
- validation

Ad-hoc activity → **Structured practice**

PI / lab / research manager

- research provenance
- knowledge retention when people leave
- traceability from publication > experiment > data
- resource/cost visibility
- reproducibility across the group

Individual activity → **Institutional research knowledge**

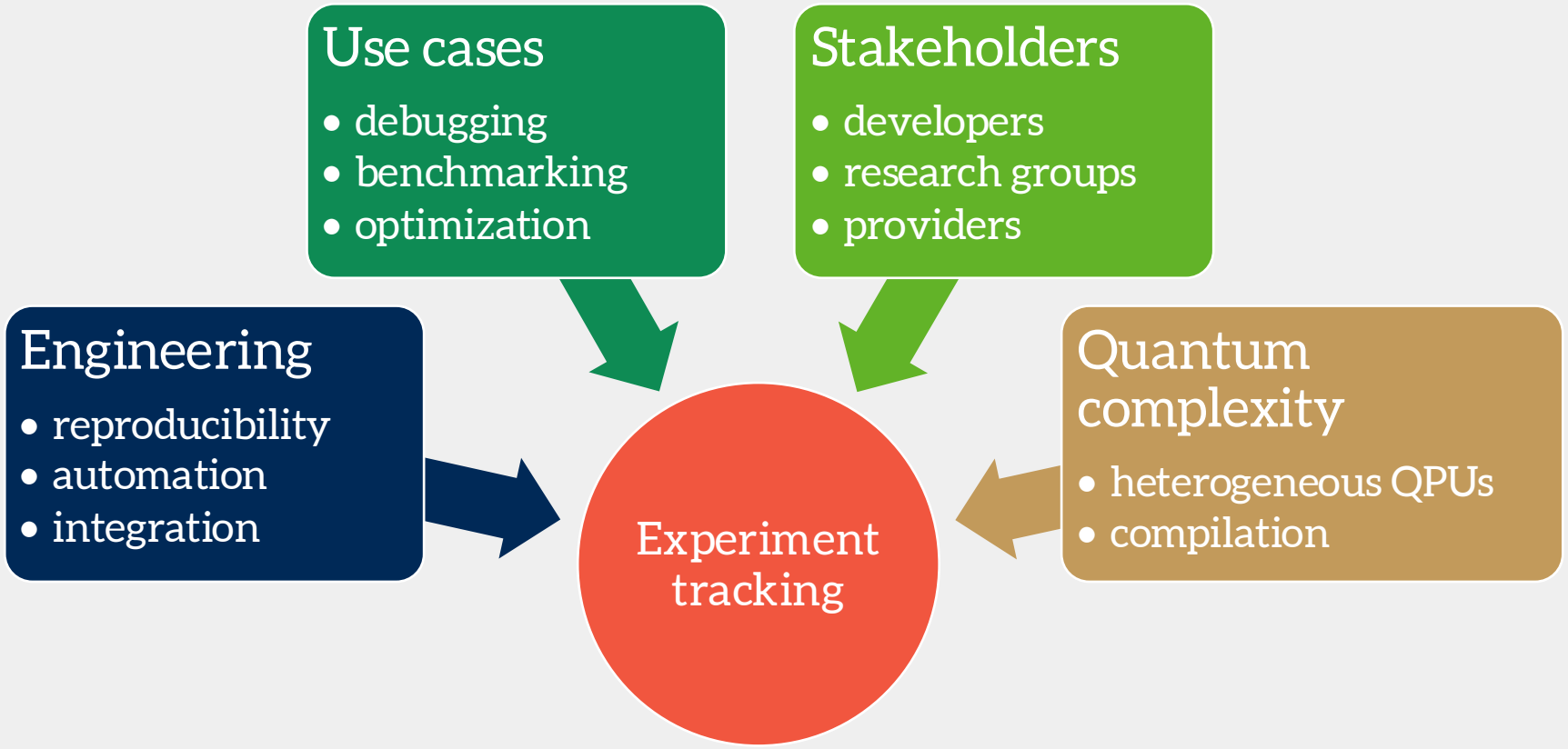
Infrastructure provider/Hardware manufacturer

- workload characteristics
- resource utilization
- failure analysis
- scheduling
- comparing simulators/QPUs
- understanding demand

- workload characterization
- performance trends
- software/hardware interaction
- calibration effects
- identifying application-level bottlenecks

Guesswork → **Transparency and visibility**

Experiment tracking as shared infrastructure



Fragmented experiment records → **Shared, structured** evidence

Approach

- Leverage best practices and tools from classical
- Not a one size fits all area, recognize differences
- Collaboration