XIII Conference on Quantum Foundations

From Foundations To Quantum Technologies

November 27th to 29th - Córdoba, Argentina

FAMAF
Facultad de Matemática, Astronomía, Física y Computación

CONICET

Quantum Computing and Its Resources in Software Engineering

Lic. Marcos Guillermo Lammers
PhD. Student in Computer Science
marcos.lammers@lifia.info.unlp.edu.ar

Lifia

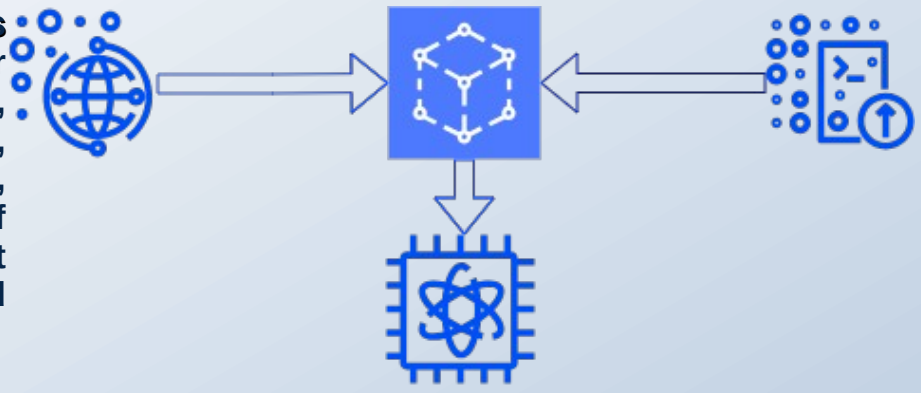
UNIVERSIDAD NACIONAL DE LA PLATA

quantum logic + probability

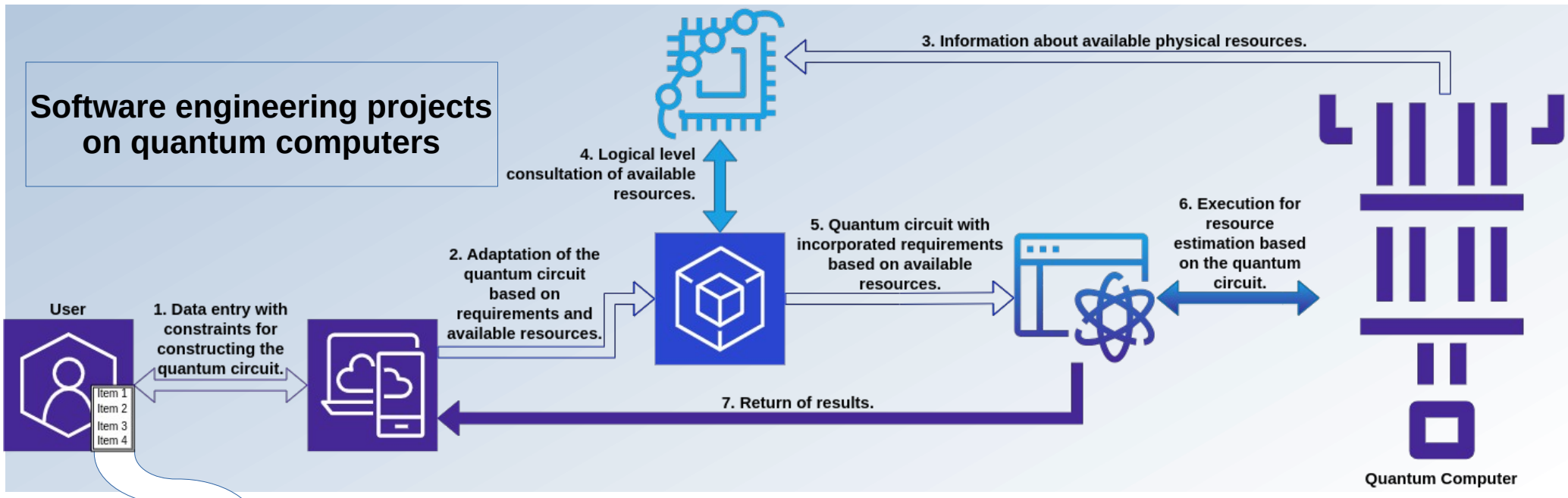
Director: Dr. Alejandro Fernández alejandro.fernandez@lifia.info.unlp.edu.ar - **Scientific Advisor:** Dr. Federico Holik holik@fisica.unlp.edu.ar

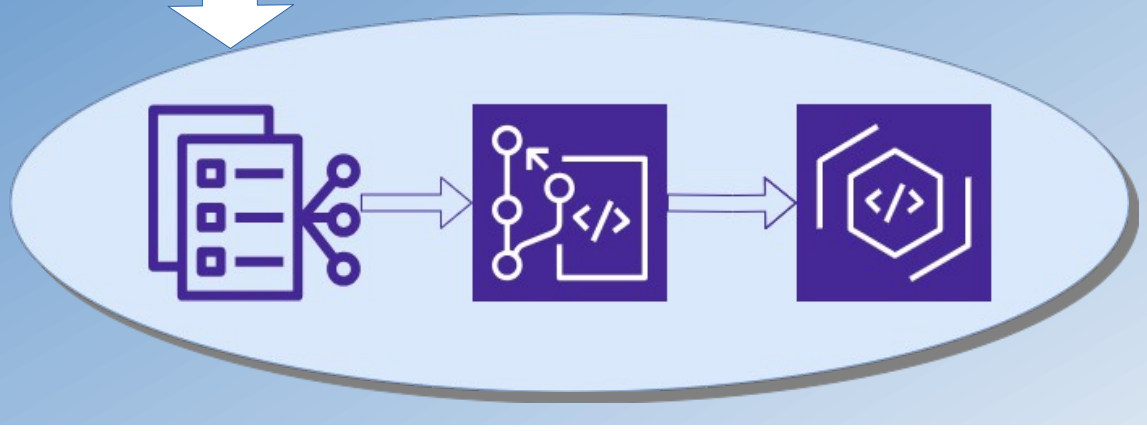
This research aims to identify and characterize quantum computing resources for software engineering, considering NISQ device limitations.

Physical resources such as the number of physical qubits, error rate, coherence time, level of entanglement, qubit connectivity, and noise level.



Logical resources such as the number of logical qubits, Clifford gates, non-Clifford gates, depth of quantum circuits, quantum algorithms, error correction protocols, quantum measurements and error estimation tools.

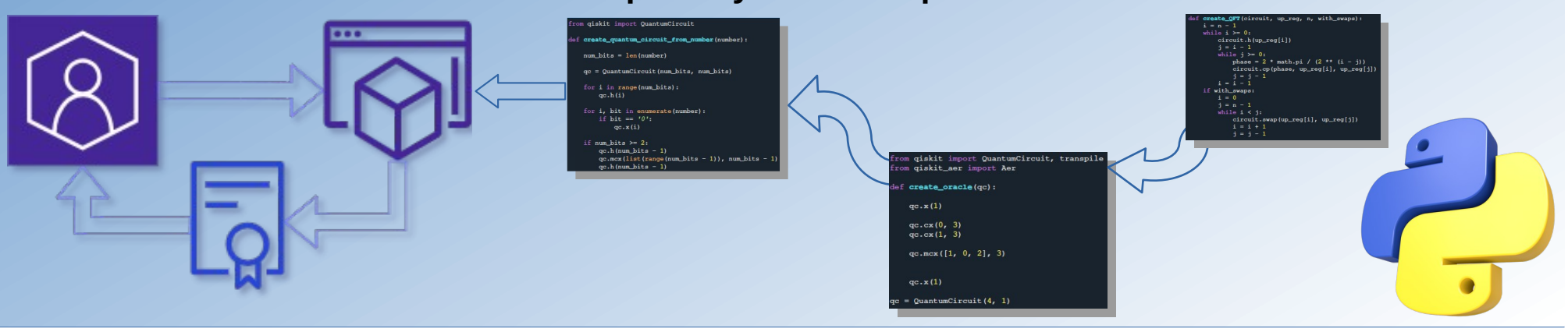




There will be a guide of steps and recommendations for estimating the resources related to the implementation of quantum software in order to use those estimates in decision-making throughout the quantum software life cycle.

Tools for assessing the utility of accessible devices.

Develop tools that help users assess the utility of accessible devices. These issues are especially relevant in the NISQ era, where publicly available quantum devices have limited resources.



```
from qiskit import QuantumCircuit

def create_quantum_circuit_from_number(number):

    num_bits = len(number)

    qc = QuantumCircuit(num_bits, num_bits)

    for i in range(num_bits):
        qc.h(i)

    for i, bit in enumerate(number):
        if bit == '0':
            qc.x(i)

    if num_bits >= 2:
        qc.h(num_bits - 1)
        qc.mcx(list(range(num_bits - 1)), num_bits - 1)
        qc.h(num_bits - 1)
```

```
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer

def create_oracle(qc):

    qc.x(1)

    qc.cx(0, 3)
    qc.cx(1, 3)

    qc.mcx([1, 0, 2], 3)


    qc.x(1)

qc = QuantumCircuit(4, 1)
```

```
def create_QFT(circuit, up_reg, n, with_swaps):
    i = n - 1
    while i >= 0:
        circuit.h(up_reg[i])
        j = i - 1
        while j >= 0:
            phase = 2 * math.pi / (2 ** (i - j))
            circuit.cp(phase, up_reg[i], up_reg[j])
            j = j - 1
        i = i - 1
    if with_swaps:
        i = 0
        j = n - 1
        while i < j:
            circuit.swap(up_reg[i], up_reg[j])
            i = i + 1
            j = j - 1
```