Quantum Software Testing Tutorial

Shaukat Ali and Xinyi Wang
Department of Engineering Complex Software Systems
Simula Research Laboratory, Norway

simula

NordIQuEst-ENCCS online HPC-QC workshop
June 9, 2022

Outline

- Introduction to software testing
- Introduction to quantum software testing
- Introduction to quantum software testing with Quito tool
- Hands on Quito and exercises

Introduction to software testing

Software is all around us

- **Airplanes:** Boeing 787 Dreamliner, approx. 6.5 million lines of code
- **Cars:** Chevy Volt, over 10 million lines of code
- **Android operating system:** 12-15 million lines
- **All Google services:** approx. 2 Billion lines of code
- …

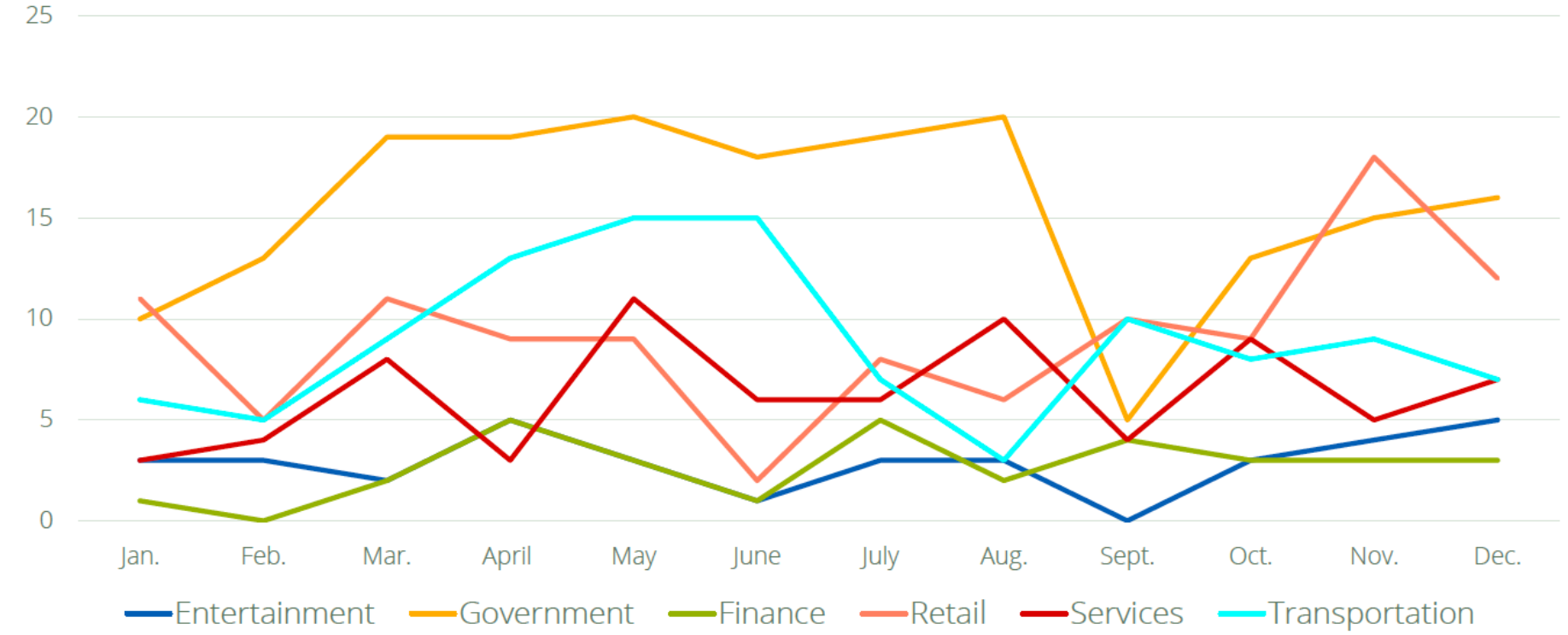
[1] https://www.visualcapitalist.com/millions-lines-of-code/

And so do their failures…

- **June 1996:** Ariane 5 disaster right after the take-off [1]
- **April 2014:** Around 11 million effected due 911 emergency services shut down [2]
- **November 2021:** Tesla recalled 12000 cars due to software error [3]
- …

[1] https://www.bugsnag.com/blog/bug-day-ariane-5-disaster
[2] https://www.theverge.com/2014/10/20/7014705/coding-error-911-fcc-washington
[3] https://www.reuters.com/business/autos-transportation/tesla-recalling-nearly-12000-us-vehicles-over-software-communication-error-2021-11-02/

Software Failure by Industry



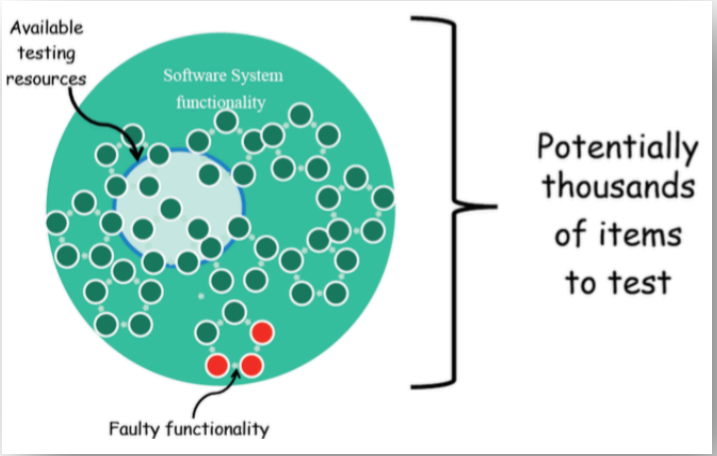
Source: https://www.tricentis.com/resource-assets/software-fail-watch-2016/

472 stories picked up by multiple news outlets

What is Software Testing?

- "Testing is the process of executing a program with the intent of finding faults" [1]
- The challenge is to identify scenarios that trigger the faulty functionality of software

"To test a program is to try to make it fail" [2]



[1] Mayers G.J. 1977. The Art of Software Testing. John Wiley and Sons, New York, USA
[2] B. Meyer, "Seven principles of software testing," *Computer,* vol. 41, no. 8, 2008.

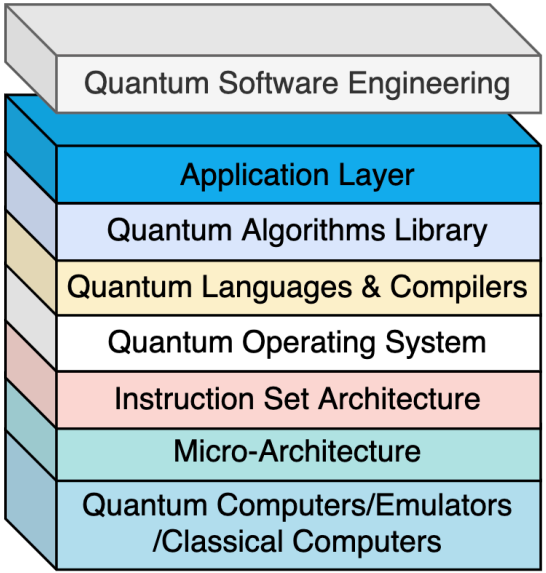
Introduction to quantum software testing

Quantum Computing (QC)

- QC promises to revolutionize the present form of computing
- QC Platforms
  - ✓ Quantum Inspire from QuTech
  - ✓ Microsoft Quantum computing platform
  - ✓ IBM Quantum Experience
  - ✓ D-Wave
- QC High level programming languages
  - ✓ OpenQL by TU Delft, Silq by ETH Zürich, Q# by Microsoft, Qiskit by IBM, Cirq by Google

It is time to build useful QC applications!

Quantum Software plays the key role in the Second quantum revolution.[1]



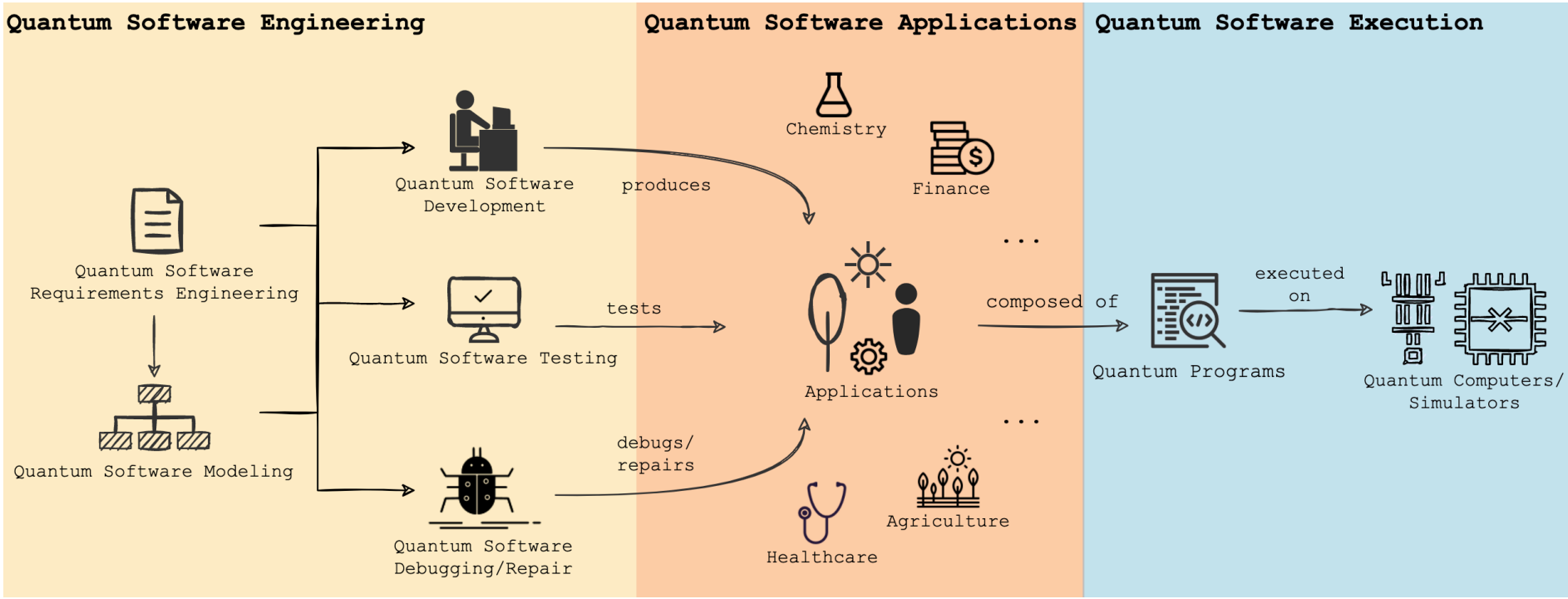
Layered QC Architecture
(Prof. Bertels's Vision)

Quantum software is at the core of the promised revolutionary QC applications.

Quantum software engineering enables cost-effective and scalable development of dependable quantum software.

[1]. Dowling, Jonathan & Milburn, Gerard. (2003). Quantum Technology: The Second Quantum Revolution. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences. 361. 1655-74. 10.1098/rsta.2003.1227.

Quantum software engineering



Thus, there is a need for novel quantum **software development methods** for developing complex quantum software-based **QC applications that are dependable**.

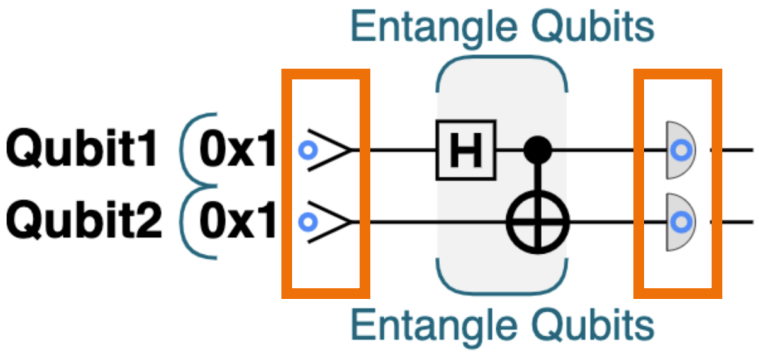
Challenges of testing quantum programs

- Probabilistic nature
- Difficulties in estimating a program's state in superposition
- Lack of precise test oracles or absence of oracles

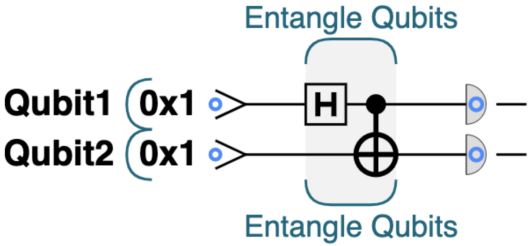
Introduction to quantum software testing with Quito tool

Inputs and Outputs of a Quantum Program (QP)

- Inputs
  - ✓Values of qubits after QP initialization
- Outputs
  - ✓Values of qubits obtained after measurement



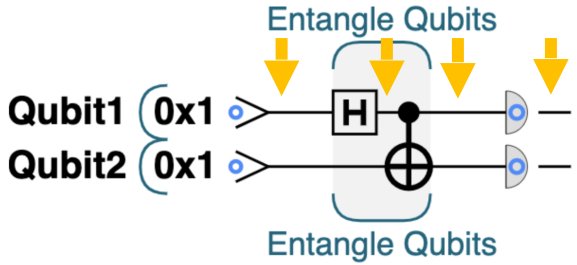
Program Specification (PS) of a QP

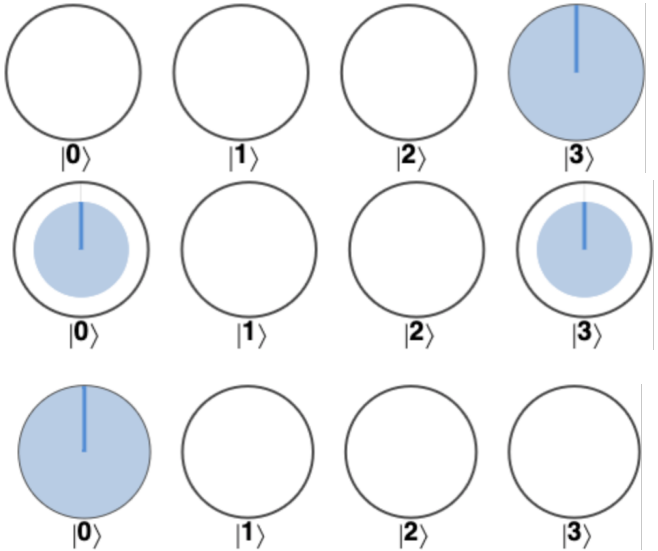


- Valid Inputs
  - ✓ Input values that are valid according to PS
- Valid Outputs Values
  - ✓ Output values that can be produced with at least one valid input
- Probabilities
  - ✓ Given a valid input, expected probabilities of occurrence of all the valid output values

Valid Input	Valid Output 1	Probability 1	Valid Output 2	Probability 2
$0 \rightarrow (00)_2$	$0 \rightarrow (00)_2$	50%	$3 \rightarrow (11)_2$	50%
$1 \rightarrow (01)_2$	$0 \rightarrow (00)_2$	50%	$3 \rightarrow (11)_2$	50%

```
1   qc.reset(2);
2   var a = qint.new(1, 'a');
3   var b = qint.new(1, 'b');
4   qc.reset(2);
5   qc.write(0);          // Initialize with 0
6   qc.nop();
7   qc.label('entangle');
8   a.had();              // Hadamard Gate. Place into superposition
9   b.cnot(a);            // Control-NOT Gate. Entangle
10  qc.label();
11  qc.nop();
12  var a_result = a.read();  // The two bits will be random,
13  var b_result = b.read();  // but always the same.
14  qc.print(a_result);
15  qc.print(b_result);
```





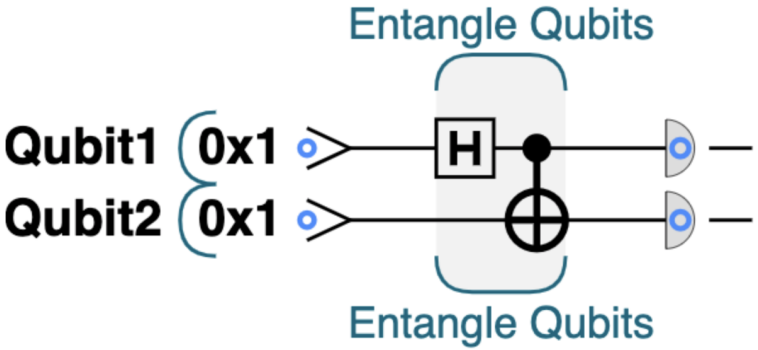
[5] M. Gimeno-Segovia, N. Harrigan, and E. Johnston, Programming Quantum Computers: Essential Algorithms and Code Samples.O'Reilly Media, Incorporated, 2019. [Online].

Quito: A Framework for Quantum Program Testing

3 Coverage Criteria	• Input Coverage • Output Coverage • Input-Output Coverage
2 Test Oracles	• Wrong Output Oracle • Output Probability Oracle
Assessment	• Mutation Operators • Mutation Analysis

Input Coverage (IC)

- In one test suite, there exists a test for each valid input
- A statically generated test suite can achieve IC



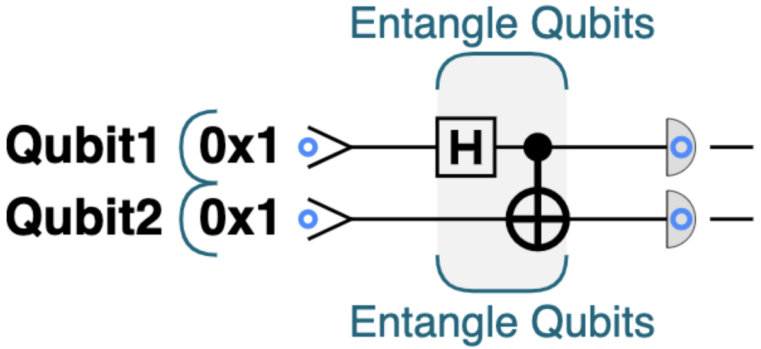
One Possible Test Suite

Input	Output
0	0
1	0

Program Specification for Entanglement

Valid Input	Valid Output 1	Probability 1	Valid Output 2	Probability 2
0	0	50%	3	50%
1	0	50%	3	50%

Output Coverage (OC)



- In one test suite, there exists a test for each valid output.
- The criterion cannot be achieved statically.

One Possible Test Suite

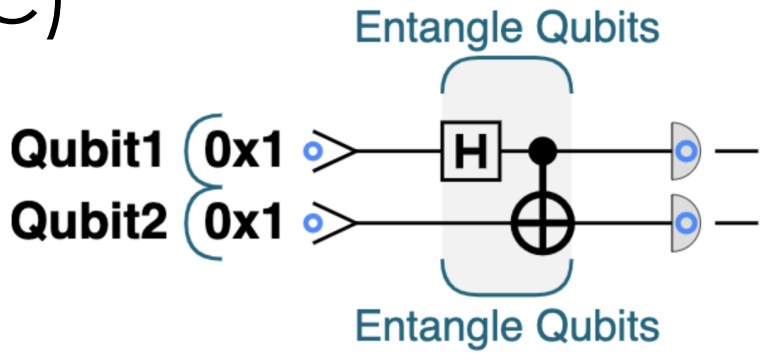
Input	Output
0	0
1	0
0	3

Program Specification for Entanglement

Valid Input	Valid Output 1	Probability 1	Valid Output 2	Probability 2
0	0	50%	3	50%
1	0	50%	3	50%

Input-Output Coverage (IOC)

- In one test suite, there exists a test for each input-output pair.
- The criterion cannot be achieved statically.



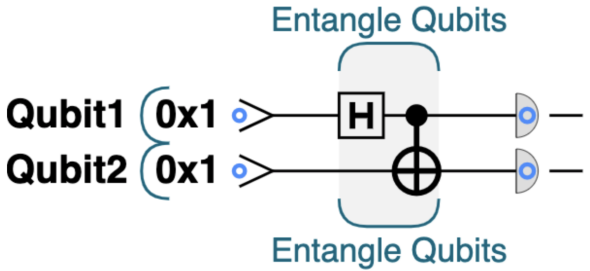
Program Specification for Entanglement

Valid Input	Valid Output 1	Probability 1	Valid Output 2	Probability 2
0	0	50%	3	50%
1	0	50%	3	50%

One Possible Test Suite

input	Output
0	0
0	0
0	3
1	0
1	3

Test Oracle – Wrong Output Oracle (WOO)



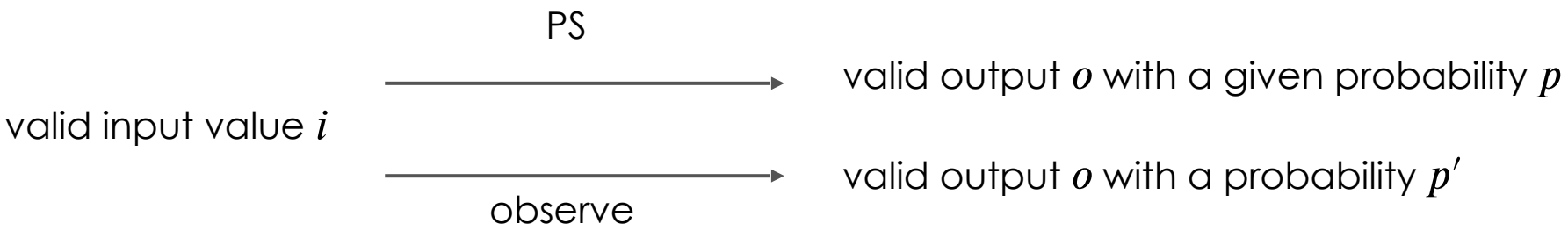
WOO checks if the test outcome returned for a test input is invalid, which reveals a *definitely fail*: wrong outputs.

Valid Input	Valid Output 1	Probability 1	Valid Output 2	Probability 2
0	0	50%	3	50%
1	0	50%	3	50%

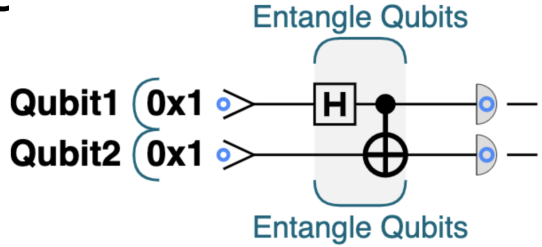
Input	Output
0	1

Test Oracle – Output Probability Oracle (OPO)

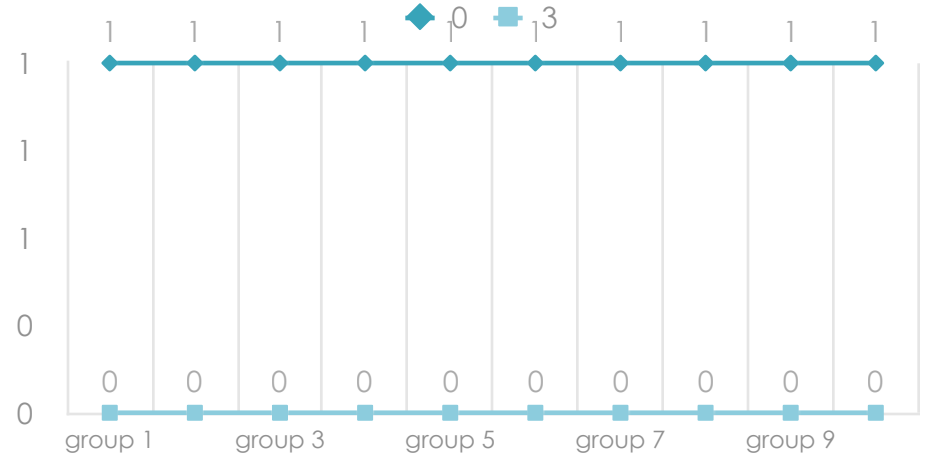
- OPO checks if a QP returns an expected output with the expected probability.
- **Likely Fail:** With a given confidence, multiple executions of a test show that the outputs do not occur with the expected probabilities.
- **Inconclusive:** Multiple executions of the test do not allow to reject the null hypothesis of a statistical test.



Test Oracle – Output Probability Oracle (OPO)

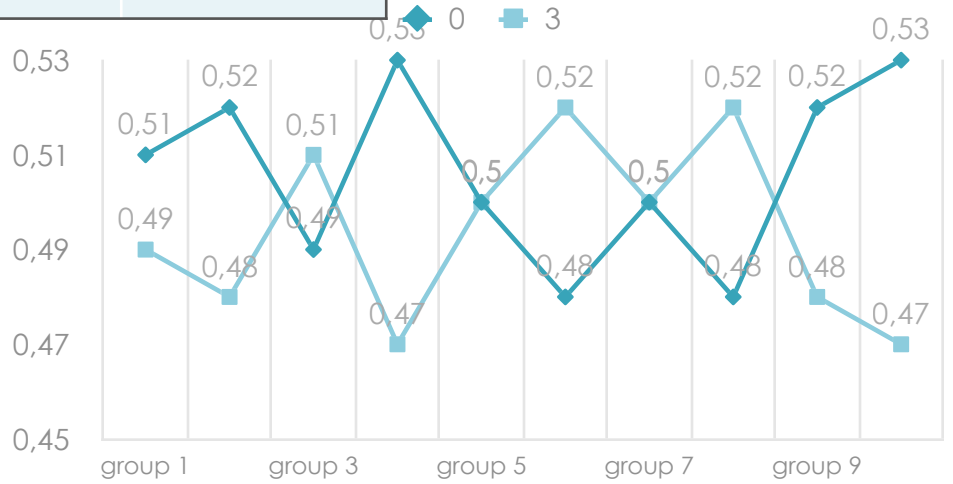


Valid Input	Valid Output 1	Probability 1	Valid Output 2	Probability 2
0	0	50%	3	50%
1	0	50%	3	50%



$(0,0)$: $p - value < 10^{-4}$
$(0,3)$: $p - value < 10^{-4}$

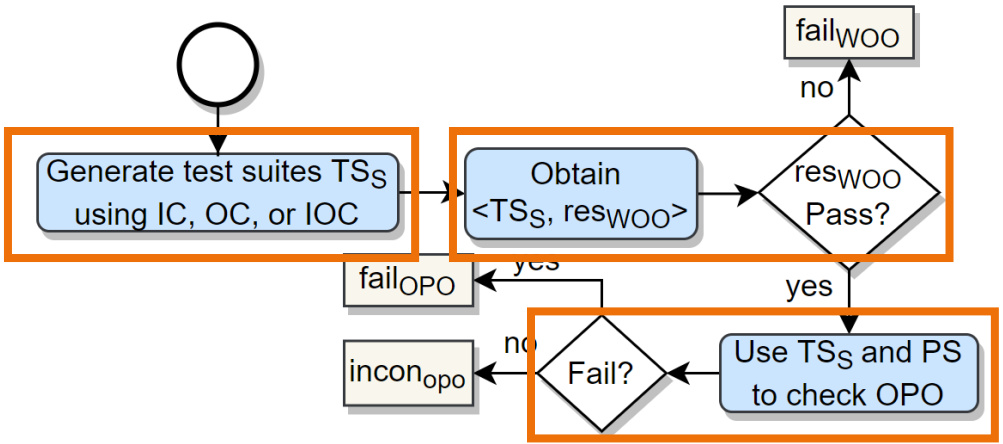
Likely Fail



$(0,0)$: $p - value > 0.01$
$(0,3)$:
$p - value > 0.01$

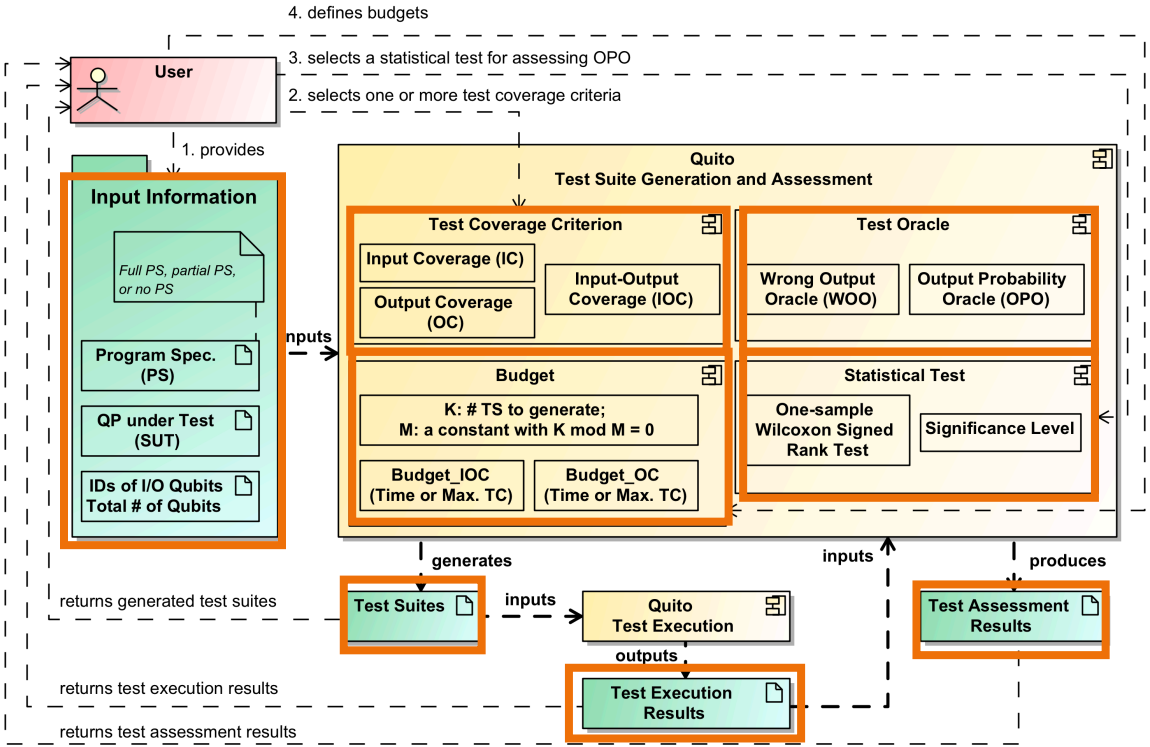
Inconclusive

Test Generation and Assessment



Hands on Quito and exercises

Overview


4. defines budgets
3. selects a statistical test for assessing OPO
2. selects one or more test coverage criteria
1. provides
User
Input Information
Full PS, partial PS, or no PS
Program Spec. (PS)
QP under Test (SUT)
IDs of I/O Qubits Total # of Qubits
inputs
Quito
Test Suite Generation and Assessment
Test Coverage Criterion
Input Coverage (IC)
Output Coverage (OC)
Input-Output Coverage (IOC)
Test Oracle
Wrong Output Oracle (WOO)
Output Probability Oracle (OPO)
Budget
K: # TS to generate; M: a constant with K mod M = 0
Budget_IOC (Time or Max. TC)
Budget_OC (Time or Max. TC)
Statistical Test
One-sample Wilcoxon Signed Rank Test
Significance Level
generates
inputs
produces
returns generated test suites
Test Suites
inputs
Quito Test Execution
outputs
Test Assessment Results
returns test execution results
Test Execution Results
returns test assessment results

Installation

- Clone the current repository git clone https://github.com/Simula-COMPLEX/quito.git
- Install **R** environment. https://cran.r-project.org
- Install **Anaconda**. https://www.anaconda.com/

Installation

- Create a conda environment (e.g., with name ” quantum_env"):

```
conda create -n quantum_env python=3.9
```

- Activate the environment and install Qiskit and rpy2

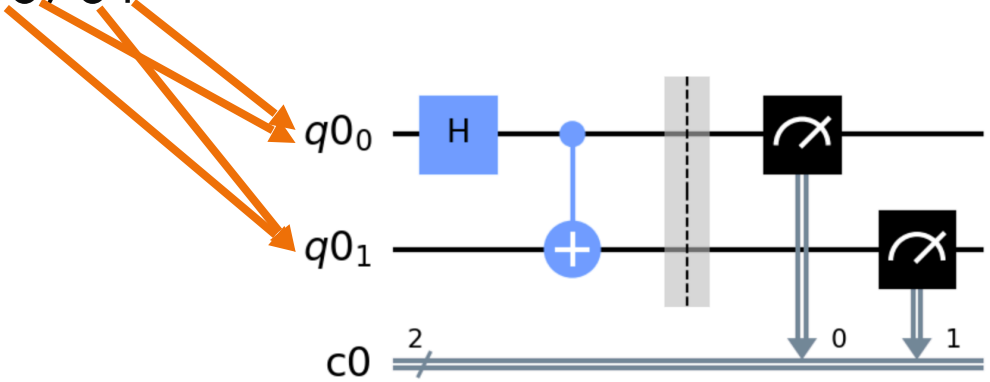
```
conda activate quantum_env
```

Example

- Program
  - ✓Entanglement
- Valid inputs
  - ✓00, 01

- Program specification

Input	Output 1	Probability 1	Output 2	Probability 1
00	00	50%	11	50%
01	00	50%	11	50%



Example

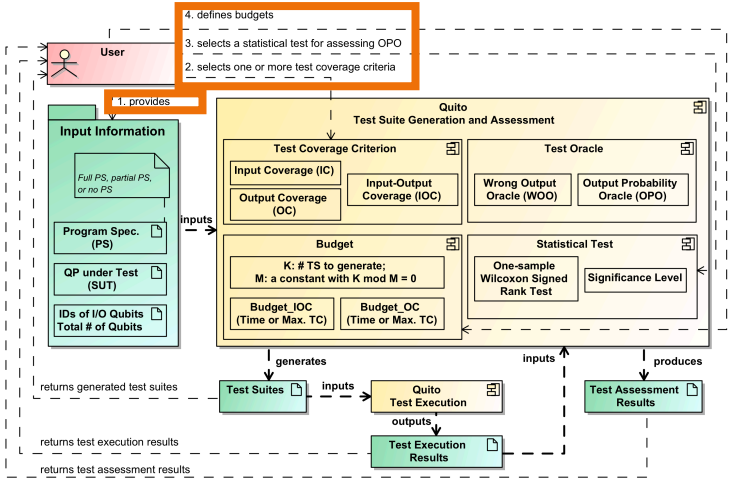
entanglement.ini

```
[program]
root=/Users/xinyi/Documents/NordIQuEst/quito-main/tutorial/example/entanglement.py
num_qubit=2
inputID=0,1
outputID=0,1

[program_specification_category]
ps_category=full

[quito_configuration]
coverage_criterion=IC
K=200
M=10
BUDGET=20
confidence_level=0.01
statistical_test=one-sample Wilcoxon signed rank test

[program_specification]
00,00=0.5
00,11=0.5
01,00=0.5
01,11=0.5
```



Example

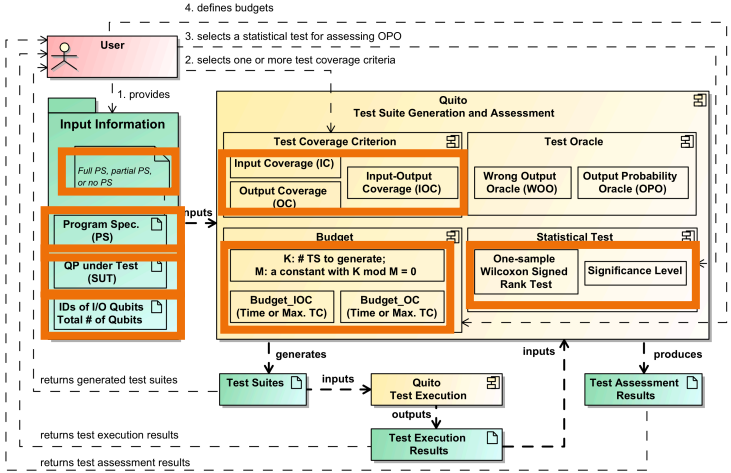
entanglement.ini

```
[program]
root=/Users/xinyi/Documents/NordIQuEst/quito-main/tutorial/example/entanglement.py
num_qubit=2
inputID=0,1
outputID=0,1

[program_specification_category]
ps_category=full

[quito configuration]
coverage_criterion=IC
K=200
M=10
BUDGET=20
confidence_level=0.01
statistical_test=one-sample Wilcoxon signed rank test

[program specification]
00,00=0.5
00,11=0.5
01,00=0.5
01,11=0.5
```

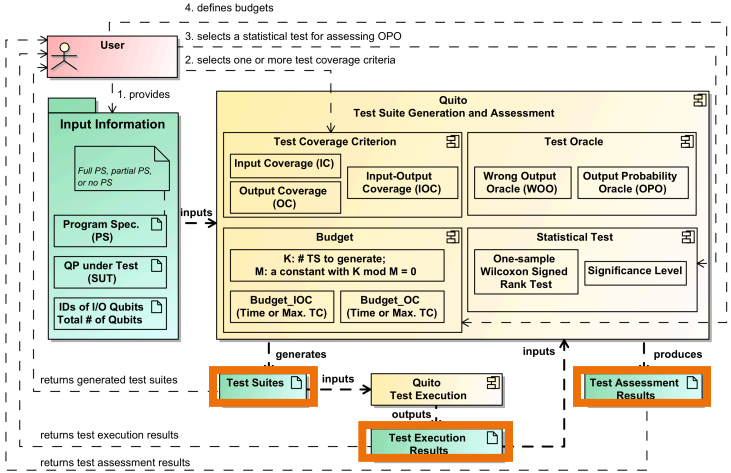


entanglement.py

```
def run(qc):
    qc.h(0)
    qc.cx(0,1)
    qc.draw(output='mpl',filename='mpl')
    qc.measure([0,1],[0,1])
```

Qiskit

Example



Test suites

ASSESSMENT_input_coverage_entanglement.txt

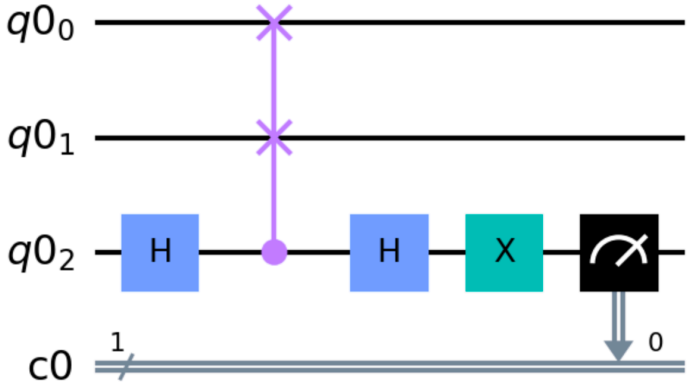
Test Execution Results

INPUTS_input_coverage_entanglement.txt

Assessment Results (for full and partial program specification)

RESULTS_input_coverage_entanglement.txt

Exercise



- Program
  - ✓Swap
- Description
  - ✓If $q0_0$ and $q0_1$ are in the same state, measure qubit will be 0 with 100% probability.
  - ✓Otherwise, there will be 50% probability for the measure qubit to be either 0 or 1.

Exercise

- Step 1
  - ✓Write quantum program file

```
def run(qc):
    qc.h(2)
    qc.cswap(2,0,1)
    qc.h(2)
    qc.x(2)

    qc.measure(2,0)
```

Exercise

- Step 2
  - ✓Write the configuration file

[program]
root=
;(Required) ;
Description: The absolute root of your quantum program file.
num_qubit=
;(Required)
;Description: The total number of qubits of your quantum program.

inputID=
;(Required)
;Description: The ID of input qubits.
;Format: A non-repeating sequence separated by commas.
outputID=
;(Required)
;Description: The ID of output qubits.
;Format: A non-repeating sequence separated by commas.

Exercise

[program_spacification_category]
ps_category=
;(Required)
;Description: The category of your program specification.
;Choice: full/partial/no

Exercise

[quito_configuration]
coverage_criterion=
;Description: The coverage criterion.
;Choice: IC/OC/IOC
K=
;(Optional)
;Description: The total number of test suites, K=200 by default.
M=
;(Optional)
;Description: The number of test suite groups, M=20 by default.

BUDGET=
;(Optional)
;Description: The budget of test suite size, BUDGET=10*number of inputs by default.
confidence_level=
;(Optional)
;Description: The confidence level for statistical test, confidence_level=0.01 by default.
statistical_test=one-sample Wilcoxon signed rank test
;(Optional)
;Description: The statistical test for assessmen
statistical_test=one-sample Wilcoxon signed rank test by default.

Exercise

[program_specification]
;(Required for full and partial program specification)
;Description: The program specification. ;Format:input string,output string=probability

Exercise

- Step 3
  - ✓Activate the conda environment

```
conda activate quantum_env
```

- Step 4:
  - ✓Start the program from the repository root

```
python Quito_CoverageRunning/quito.py
```

- Step 5:
  - ✓Enter a number to select the operation and follow the instructions

Exercise

- Mutant

```
def run(qc):
    qc.x(0)
    qc.h(2)
    qc.cswap(2,0,1)
    qc.h(2)
    qc.x(2)

    qc.measure(2,0)
```