Issue Date:	**2021-26-01**	**CEN-CLC/FGQT**	**N074**
Deadline:	**2021-11-02**	**Supersedes: N050a**	
Status:	FOR APPROVAL		

TITLE	**Changes to quantum computing use-case**
PROJECT	FGQT Roadmap
REFERRING TO	N020a, N050a
SOURCE	Giovanni Frattini (Engineering Ingegneria Informatica S.p.A), Antonello Corsi (Engineering Ingegneria Informatica S.p.A.), Michele Amoretti (CINI), Niels Neumann (TNO)
CONTACT	Giovanni.Frattini@eng.it, niels.neumann@tno.nl

Supported by:

- Rob van den Brink (Delft Circuits, The Netherlands),
  - Rob.vandenBrink@Delft-Circuits.com,
  - https://delft-circuits.com/
- Giovanni Frattini and Antonello Corsi (Engineering Ingegneria Informatica S.p.A., Italy),
  - Giovanni.Frattini@eng.it and Antonello.Corsi@eng.it
  - http://www.eng.it

ABSTRACT

Explanation of changes

We propose some changes to the agreed upon contribution N050a. The changes focus on the seamless integration of quantum computing and classical computing in the cloud. We propose to include these changes in contribution N050a in the roadmap document. Specifically, we propose to add this contribution to section 5: *Innovation and use cases*.

Why are the changes needed?

This document provides a use case for using cloud-based quantum computing. With respect the N050a this document add more details on how to seamlessly integrate, in the limit of the today evolution of the Quantum Computing technologies, traditional clouds (and related commercial services) with commercial Quantum Computing units. The sketched use-case is focused already on the near-term, where specific problems can be solved more efficiently by using quantum computers.

Instructions for editor

Section 4 to be included as-is in section 5.1.2 *Domain-specific use cases* of N020a.

1. Situation sketch

Quantum computing has the potential to solve some computational problems efficiently or provide meaningful gains in other problems. Examples of the former are cryptographic related problems, such as prime factorization and the discrete-log problem, and simulations of chemical processes. An example of the latter is machine learning, as quantum computing can provide meaningful gains in the running time, the number of training steps required or in the capacity of associative networks.

It is expected that, at least for the foreseeable future, quantum computers will mainly be hosted on few locations worldwide, which can be accessed remotely. Currently, this means that you have to log on to some programming environment of the host and program your quantum algorithm there. Another option is that you program the quantum instructions locally and then send everything to the host. Ideally however, users are not ~~bothers with~~interested in technicalities of implementations and low-level quantum instructions, but instead only with the result of a quantum routine.

An example of a high-level quantum computing software-stack is shown in Figure 1. Currently, the separation between a local user and a remote quantum host is high in the stack. Already on a high level, instructions must be programmed or shared with the quantum computer host. Therefore, users are required to program their algorithms in low-level instructions.

The desired separation is however lower in the stack, such that users can program quantum instructions on a high-level and function libraries can be used when programming the quantum or hybrid algorithm.

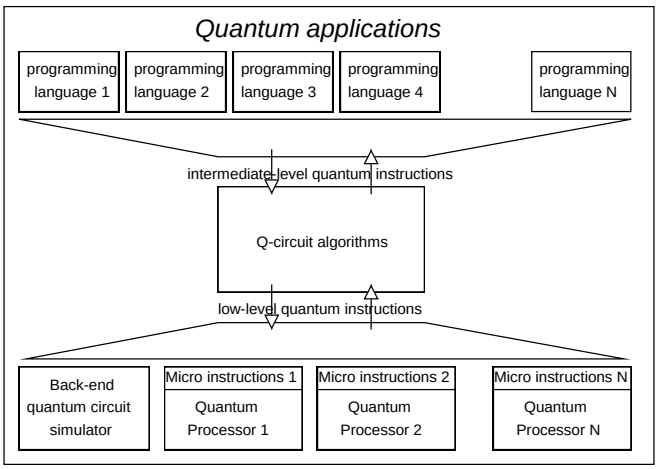

Figure 1: A software stack for quantum computing (from: Van den Brink, Neumann, Phillipson "Vision on Next Level Quantum Software Tooling", 2019).

2. Use case of quantum computing

Suppose a user has developed an algorithm focussed on pattern-recognition with a specific computationally heavy sub-routine. A hybrid algorithm might provide the solution, however this requires the algorithm to be partly run on a classical cloud (and/or a supercomputer, high-performance computing (HPC), high-throughput computing (HTC)) ~~(super)computer~~ and partly on a quantum computer. The access to Quantum resources, them being distributed across the globe, needs communication networks (e.g., fiber optical communication channels). The business models are going to be based on parameters such as requested Service Level Agreement (SLA), time for executing the algorithms, intellectual rights (IP), etc.~~As quantum computers are likely to be hosted in the cloud and quantum computing time can be bought.~~

In this case, there are two ~~points of~~ major attention points:

1. The user will likely not want to share ~~their~~ its secrets (its data and possibly also the algorithms running on the remote quantum host, if protected by IP);~~algorithm;~~
2. The user is usually only interested in the result of the sub-routine, rather than in the way it is executed.

It could be also given the case in which the user just uses existing and proven Quantum algorithms asking for the highest level of security against eventual information flaws.

Currently, the user has to program ~~their~~ its algorithm in low-level quantum instruction and provide those to the host, or program the algorithm in an environment provided by the quantum host. Nevertheless, we expect an evolution of the quantum software engineering that will lead to a convergence among Quantum Programming environments and software life cycles with those of traditional clouds. We expect, then, that~~Instead,~~ the user should be able to program as high-level as possible, adopting a unique, homogenous software development process. Therefore, a quantum library~~,~~ with high-level quantum instructions is essential. Instructions in this library are then compiled to lower-level instructions. Quantum instructions, once compiled, could be retained locally, transferred on the quantum remote host or, possibly, deployed into "containers", more manageable modules, whose management is then assimilable to the ones adopted by all the major cloud providers across the globe. ~~The library can be hosted at the user-side or at the host-side, and similarly for the compilation of the instructions.~~

It is envisaged a direct integration with classical programming languages: in this case the underneath integration among quantum and classical computing resources would be masked to the user. Therefore, the user could run its program not caring about the complexities that the execution of instructions on the underneath quantum hardware, entails.
~~This quantum library should be callable from most to all often used classical programming languages.~~
~~The user should be able to opt for running the quantum library locally if the algorithm is confidential or the user is not willing to share the algorithm for other reasons.~~ Both the low-level instructions and those supported by the quantum library should at least support a standard instruction set. Note that these standardized low-level instructions are hardware-agnostic. In a compilation step, these instructions can be translated to hardware-specific operations, based on the used hardware-backend.

The user should be able to opt for protecting its algorithm implementation based on the high-level quantum library, by compiling it to the low-level instruction set in local enclaves, protected from indiscrete eyes and theft of Intellectual Property. The user should also be able to preserve the secrecy of the compiled program, by means of indistinguishability obfuscation (iO). iO requires that if two programs P0 and P1 are two implementations of the same function, then their obfuscations are computationally indistinguishable. This is not the strongest notion of obfuscation (which is denoted as *virtual-box obfuscation*, and is known to be impossible for general-purpose functionalities), but yet the best possible one.

It is important to underline that the intermediate artefacts of the quantum computation are not business relevant. The ~~Furthermore, the~~ compiled quantum instructions are usually not of interest to the user, ~~instead~~while~~, only~~ the end-result is very relevant to the user. More in general, the quantum-technology could be relevant if and only if a certain SLA is established among the parties: the target technology could be then decided upon automatically, based on requirements imposed by the agreement among the parties, the algorithm and the business constraints.~~Therefore, also the used backend quantum-technology is in general not of interest and could be decided upon automatically, based on requirements imposed by the algorithm.~~ Likewise~~Finally~~, error-correcting methods should be applied if needed, without users having to worry about them. Note that it should be possible for users to indicate a preference for specific backends, for instances as user agreements may differ between different quantum computer hosts.

As multiple users may want to use the quantum computer simultaneously, there is a need for protocols and scheduling mechanisms, for matching request and demand of computational services. Last but not least, Quantum computational services must be accounted, billed and invoiced according to the service required~~, billing and job-control to facilitate who runs first and for how long~~. These features should be provided by suitable "operating systems" enabling a seamless interaction~~For a specific run, the user should be able to interact~~ with the cloud-based quantum computers~~ on what instructions to execute, but also when the calculation is finished and what the results are.~~

Additionally, the classical part of the algorithm is likely to be run on a high performance computer (supercomputer). Therefore, the supercomputer must be able to interact with the quantum computer, sending instructions and retrieving results.

There are two important notes on the above:

~~1.~~ The standardized low-level gate set should be universal and hardware-agnostic. By high-level compilation to this standardized low-level gate set, it can in a second step be mapped to arbitrary hardware-backends. This mapping constitutes translating the low-level gates to hardware-specific operations, based on the used hardware-backend.

1.

2. Current quantum-hardware options require users to access the devices from a remote environment. The term cloud in this document relates to how quantum computers should be used in the future and has similarities with the classical cloud.

~~2.~~3. Requirements for quantum computing

The ~~example~~ use-case presented above sketches various requirements for quantum computing to be practically used:

- A ~~local~~ hybrid programming environment where classical and quantum instructions can be used;
- Software and APIs for seamless integration of quantum jobs into complex business services;
- Support for high-level quantum instructions;
- Compilation of these high-level quantum instructions to a set of standardized (hardware-agnostic) low-level instructions (e.g. QASM) and some specific ~~higher~~ intermediate-level instructions (e.g. quantum Fourier transform);
- Both local and remote support for compilation. The user may opt to compile locally, or may have no preference;
- Containerization of Quantum services, allowing a seamless management of Quantum software in relevant industrial contexts. Containers should expose interfaces (according to a Service Oriented Architecture approach) callable from remote hosts, having full control over those components;
- A software stack abstracting Quantum resources, acting as a broker for Quantum resources;
- In case of (hardware-agnostic) low-level quantum instructions ~~send~~ sent by the user, there should be the option for integration of hardware specific constraints;
- Support for indistinguishability obfuscation of (hardware-agnostic) low-level quantum instructions sent by the user;
- In case of high-level instructions~~ send by the user~~, there should be adequate messaging to facilitate debugging;
- Protocols on how to interact with the quantum computer, send instructions, ~~and~~ retrieve results, manage jobs and possibly quantum modules (containers) in larger software service contexts mixing traditional and quantum computing resources spread across traditional networks;
- Software abstracting and controlling executions, exposing interfaces (and protocols) to that allow control on quantum jobs and would help in accounting and billing Quantum resource usage according to the service (and associated SLA) bought;

- ~~Protocols on billing and job control to determine who runs first and for how long;~~
- ~~Compilation of quantum instructions to (hardware-agnostic) low-level instructions such that reconstruction of the original algorithm is hard;~~
- Translation of (hardware-agnostic) low-level instructions to hardware-specific elementary operations;
- Automatically apply error-correcting techniques (the necessity of these may depend on the specific backend and the specific SLA);
- An interface between the quantum computer and supercomputers, to send instructions and provide (intermediate) results.

~~3.~~4. To be approved text

Editor's note: The following text is to be approved as-is for section 5.1.2: Domain-specific use cases. *The changes are incorporated in the text of the agreed upon contribution N050a.*

Use Case: Using a Quantum Computer as Secondary Processor in the ~~Qloud~~Cloud

A user has developed a pattern-recognition algorithm with a specific computationally heavy sub-routine. To optimize both the running time and the results, the user wants to run part of its algorithm on a quantum computer without ~~revealing the propriety algorithm~~disclosing it. The rest of the algorithm is programmed in an often-used classical programming language and it is run on a classical cloud (or ~~(~~super~~)~~computer). Only the result of the quantum-subroutine is needed in the rest of the algorithm. The user can either run the software on a new generation quantum computer, mixing quantum and classical computing facilities or optimising the execution of the software by distributing it over the available resources. In both cases, the user does not want to deal with the selection of the most appropriate mix of computing resources. The user wants to achieve business-relevant results, while taking care to the overall cost for maintaining the software, thereby maximising its competitive advantage.

This requires a quantum-host and infrastructure, such that the user can run the quantum-subroutine (possibly wrapped into classical interfaces exposing it over a traditional network) under the requirements that

- The user can program the algorithm locally in a classical environment using a high-level classical programming language and additional quantum instructions;
- The quantum instructions can be programmed both on a high- and a low-level;
- The high-level quantum instructions are compiled to hardware-agnostic low-level instructions, either locally or at the host. The ~~company~~ user should be able to decide on this;
- The low-level quantum instructions should be from a hardware-agnostic standardized gate set;
- If the user decides to compile locally, low-level instructions are ~~send~~ sent to the cloud-based quantum computer. The user should then also have the option to integrate hardware specific constraints. Additionally, this should be taken care of by the host if needed;
- The user should have the possibility to apply indistinguishability obfuscation to the hardware-agnostic low-level quantum instructions before sending them to the cloud-based quantum computer;
- The hardware-agnostic low-level quantum instructions should then be translated to hardware backend-specific instructions~~, corresponding to the backend~~;
- If the user instead sends high-level instructions to the host, and the instructions are compiled on the host-side, there should be messaging that facilitates debugging. Additionally, the user should be able to integrate specific hardware-constraints. Otherwise, the compiler should take care of this;

- The high-level quantum instructions may in this case be directly compiled to hardware-specific instructions;
- The intellectual property of new Quantum algorithms should be protected~~It is hard to reconstruct the original algorithm from the compiled instructions~~;
- There is a standardized interface allowing for executing (bursts of) quantum instructions, and the algorithm should automatically call the quantum instructions when needed;
- Protocols exist such that the user can use it for sending their quantum instructions to the hardware whenever needed~~, :~~ instructions are processed ~~and run~~ and results are send back to the user;
- Execution of Quantum software in orchestrated by an intermediate software layers (classical) exposing standard interfaces and protocols. This operating system should allow for managing quantum jobs, auditing and billing according to requested SLAs and business relevant parameters (e.g. priorities, execution time, etc.);
- ~~Protocols exist on billing and job control to determine who runs first and for how long among different users;~~
- Quantum error-correction routines are applied if needed, the user should have influence on this if desired;
- An interface exists between the quantum computer and (super)computers, potentially hosted in the cloud, to send instructions and provide (intermediate) results;
- The result of the quantum-subroutine can directly be used by classical clouds or ~~the (~~super~~)~~computers running the whole algorithm;~~.~~
- The quantum routine should be integrated in complex service lifecycles as implemented by the users on traditional cloud;
- Such an integration is aimed to reduce costs in the medium term.