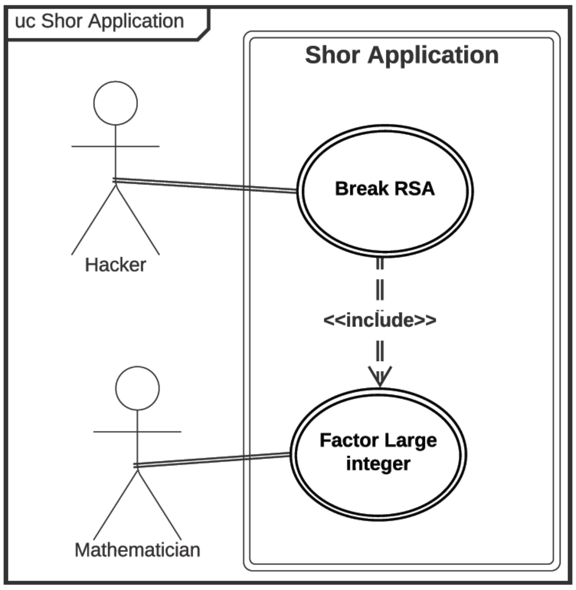
General Notes

- Carlos employs the use of UML frames in the QUML diagrams throughout the paper (https://www.uml-diagrams.org/frame.html, https://developer.ibm.com/articles/the-sequence-diagram/)
- A frame has a frame heading and contains the contents of the diagram in question
- It was introduced in UML2
- A frame is an optional boundary for the diagram in UML. It does not determine the nature of the diagram, the contents of the diagram does this, it labels the type of diagram and creates a clear boundary.
- The “Shor Application” used throughout Carlos’ paper is an implementation of Shor’s algorithm using quantum software.

Use Case

Summarises the interaction between a system and it’s users - https://www.educative.io/answers/what-are-include-and-extend-relationships-in-a-use-case-diagram

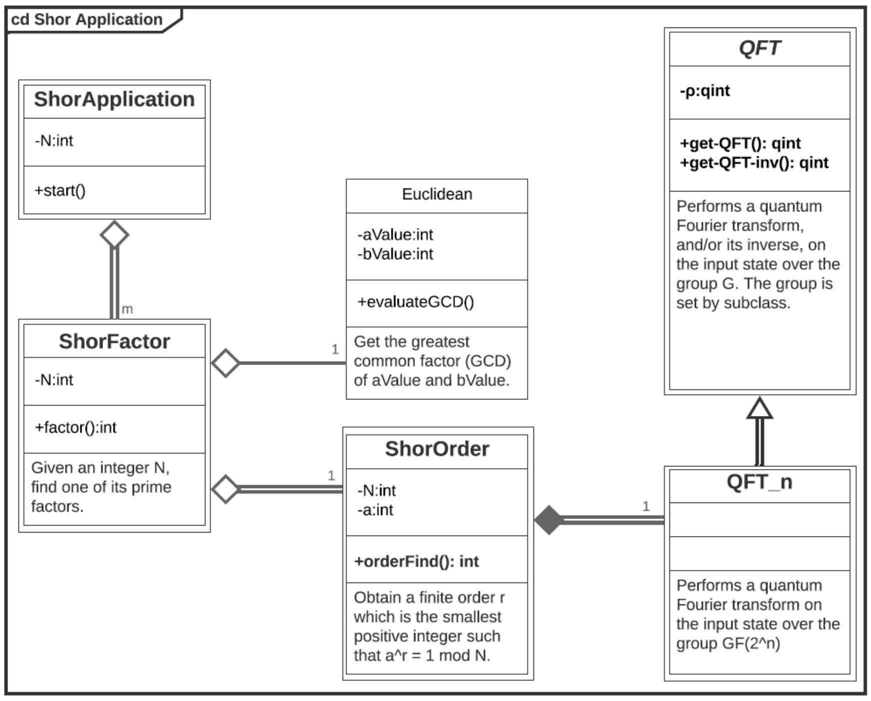


- When using Miro to create the diagrams, BPMN needs to be used to create squares or circles with double lines
- Page 25 of UML @ Classroom “This rectangle symbolises the boundaries of a system to be described”. This is the rectangle that encapsulates the Shor Application system used for example.
- Page 24 of UML @ Classroom “It encompasses a number of functions used when using the system…a use case is triggered by an event or an actor…can be used to document the functionality that a system offers. It is usually represented by an ellipse”. The ellipse within the rectangle shows two functions of shor's algorithm, one being to break RSA, the other being to Factor Large Integer

- <<include>> ; The use case is mandatory and part of the base use case. It is represented by a dashed arrow in the direction of the included use case. In our case, you have to factor large integers in order to break RSA. It points from the base use case to the included use case. https://www.educative.io/answers/what-are-include-and-extend-relationships-in-a-use-case-diagram
- With our two actors, a hacker will want to use the functionality “Break RSA” in order to hack into other systems, and the Mathematician will want to use the “Factor Large Integer” functionality for their equations.
- Carlos states in his paper the use of double lines is to denote the use of quantum resources by the software system. The actors are each using a quantum resource as they are interacting with a quantum system, hence double lines here depicting each actor triggering the use cases they would be interested in. The double lines around the use cases depict they are quantum functions, and the double lines around the rectangle denote it as a quantum system.
- Bold font is also used to depict quantum systems, with the font for the rectangle (system) and use cases (functions) in bold. The actors are not in bold as they are not quantum systems.

Class Diagram

A static diagram illustrating the relationship between classes in a system and defining attributes and methods of said classes - https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/



cd Shor Application
ShorApplication
-N:int
+start()
m
ShorFactor
-N:int
+factor():int
Given an integer N, find one of its prime factors.
Euclidean
-aValue:int
-bValue:int
+evaluateGCD()
Get the greatest common factor (GCD) of aValue and bValue.
1
ShorOrder
-N:int
-a:int
+orderFind(): int
Obtain a finite order r which is the smallest positive integer such that a^r = 1 mod N.
1
QFT
-p:qint
+get-QFT(): qint
+get-QFT-inv(): qint
Performs a quantum Fourier transform, and/or its inverse, on the input state over the group G. The group is set by subclass.
QFT_n
Performs a quantum Fourier transform on the input state over the group GF(2^n)
1

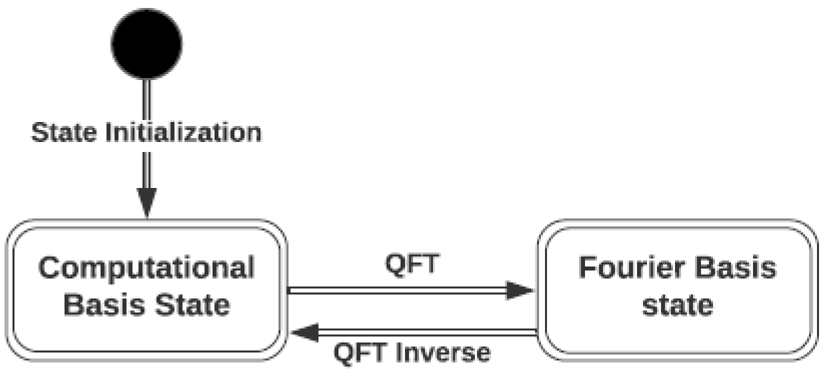
- There are 6 classes in total in this example, 5 being quantum and 1 being classical. Quantum classes are denoted in bold text for the class name and a double lined border for the rectangle. Classical classes (Euclidean) have single border boxes. Page53 of UML@C states in UML the class name is normally in bold font, which is not the case in Carlos' design. It may be the case that classical must change to non-bold, a design change with UML, when using QUML. This needs to be looked into further.

- The class ShorApplication has an attribute "-N:int" and an operation "+start()". The attribute specifies information contained for all instances of the class but each instance may have a different value. (UML@C page 53). The operation specifies what actions can be performed by instances of the class (google gemini)
- The class is broken up into multiple compartments; the first being the name of the class, the second compartment being attributes, the third being operations.
- -N:int broken down; - is private visibility (access only within the object itself permitted), N is the name of the attribute and :int is its type, integer.
- +start() broken down; it is a public method (permitted by instances of any class) which starts the Shor algorithm in our quantum system.
- A shared aggregation is depicted between the ShorApplication class and the ShorFactor class. An aggregation is a type of association where there is communication between two classes. In shared aggregation, there is a relationship between the two classes but the child class can still act independently of the parent class. (https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/) In our example, ShorFactor is the child class and has an aggregation with ShorApplication. This implies that the ShorFactor class can still find a prime factor of an integer and does not depend on ShorApplication to do this, however there is a relationship between the two classes where ShorApplication will initialise the process of factoring prime numbers with its start() method. The letter "m" is used at the child end of the aggregation, this isn't standard UML notation and likely represents many which is noted as * , this indicates that there are many instances of the ShorFactor class that can be called when the ShorApplication class executes its start method.
- The double lines of the aggregation indicate that it is quantum relationship and not a classical relationship, meaning that the two classes must communicate via a quantum channel and cannot communicate via a classical channel
- This leads nicely on to the euclidean class which is our only classical class in the system. It has a classical aggregation between itself and ShorFactor, with euclidean being the child class and ShorFactor being the parent class. They communicate via a classical channel and do not need to communicate via a quantum channel
- The Euclidean class has two private attributes; it takes two integer values. Its public operation is to find the greatest common factor between the two values. The aggregation between Euclidean and ShorFactor indicates that one instance of the Euclidean class is called when ShorFactor executes.
- ShorFactor also has an aggregation with the quantum class ShorOrder. This class has two private attributes also, N and a, both integers, with a public method orderFind() again of type integer. This method "obtains a finite order r which is the smallest positive integer such that a to the power of r = 1 mod N." One instance of the ShorOrder is called when the ShorFactor class executes its method.

- The ShorFactor class method takes an integer N and finds one of its prime factors. This process is done by calling one instance of the Euclidean class and one instance of the ShorOrder class.
- ShorOrder has a quantum composition relationship with the QFT_n class. One instance of QFT_n is called when ShorOrder executes its orderFind method. The composition means the child (QFT_n) cannot exist without the parent (ShorOrder).
- QFT_n is a subclass of the superclass QFT, as depicted by the generalisation relationship (inheritance) which is the follow arrow pointing from a subclass to a superclass. This means that QFT_n inherits the characteristics (attributes and operations) of QFT. It is depicted by a double line meaning that this is a quantum relationship, the superclass is quantum which would result in a quantum subclass
- QFT has a private attribute p:qint and two public methods, get-QFT(): qint and get-QFT-inv(): qint, the description states “Performs a quantum fourier transform, and/or inverse, on the input state over the group G. The group is set by subclass.
- Finally, we note that the description in QFT_n states “Performs a quantum Fourier transform on the input state over the group GF(2^n)” which sets the group G.
- QFT is an abstract class hence the italics and bold font. An abstract class means there is no direct instances of it and it is to be used solely as a base class for the subclass instances, an umbrella that contains the characteristics all subclasses would share - https://www.ibm.com/docs/en/zos/2.4.0?topic=only-abstract-classes-c
-

State Diagram

A diagram that models the possible states of a system or object within the system.



- The example is focusing on the object QFT used in the previous class diagram
- The two states are Computational Basis State and Fourier Basis state; both depicted as quantum states
- They have quantum operations in order to change their state, one being QTF, and one being QTF Inverse.

Activity Diagram