Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

# QuCSplit: A Decision Support System for Quantum-Classical Splitting

Mohamad Altaweel

| | |
|---|---|
| **Course of Study:** | Software Engineering |
| **Examiner:** | Prof. Dr. Dr. h. c. Frank Leymann |
| **Supervisor:** | Daniel Vietz, M.Sc. |
| **Commenced:** | May 11, 2022 |
| **Completed:** | November 11, 2022 |

# Abstract

Quantum computing is a new technology that depends on quantum mechanics concepts to solve traditional problems in computer science in different domains such as optimization, simulation and machine learning computations. The benefit of quantum computers over classical ones is that quantum algorithms provide an exponential speed-up over classical algorithms. This extraordinary promising performance motivates vendors and researchers to develop solutions using quantum computing and integrate them to solve different use cases. However, quantum computers depend on a different implementation than classical computers by using quantum circuits and data preparation and do not handle classical bits by central computations. Thus, we need to revise the translation process of high-level requirements and how to implement it as quantum solutions. For this reason, the integration and deployment of quantum computers might be expensive and not all problems can not benefit from it. Thus, In this early stage, stakeholders, e.g. researchers and enterprises, need support in this process to decide whether to use quantum or classical computers. To address this problem, we present a concept of using a decision tree to help users in this process. The decision tree is based on results that have been reviewed from literature which started to use quantum computers and compare their performance with classical methods. It consists of a set of decision which is connected consequently, starting from describing the use case requirements and transforming them into a mathematical formulation which can be solved on quantum computers. In addition, this project investigated the content comparison of the different use cases using NLP methods to get similar cases together and provide a recommendation based on similar content. The decision tree must be accessible in an easy way to all users from different backgrounds, including researchers, technical, vendors, and decision-makers in the business domain. To validate the practical feasibility of our concept, we introduce QuCSplit as a support framework to help users make their decisions. The decision tree approach can provide a reliable model of linear decisions that are sequentially ordered. However, it is still difficult to include decisions that require complex calculations and relationships. Furthermore, the texts comparison method still seeks a reliable evaluation study to analyze its accuracy.

# Kurzfassung

Quantencomputer ist eine neue Technologie, die auf Konzepten der Quantenmechanik basiert, um traditionelle Probleme der Informatik in verschiedenen Bereichen wie Optimierung, Simulation und maschinelles Lernen zu lösen. Der Vorteil von Quantencomputern gegenüber klassischen Computern besteht darin, dass Quantenalgorithmen einen exponentiellen Geschwindigkeitszuwachs gegenüber klassischen Algorithmen bieten. Diese außerordentlich versprechende Leistung motiviert Anbietern und Forschern, Lösungen auf der Grundlage von Quantencomputern zu entwickeln und sie zur Lösung verschiedener Anwendungsfälle zu integrieren. Quantencomputer ist jedoch auf eine andere Implementierung als klassische Computer angewiesen, da sie Quantenschaltungen und Datenaufbereitung verwenden und klassische Bits nicht durch zentrale Berechnungen verarbeiten. Daher muss man den Übersetzungsprozess von High-Level-Anforderungen und deren Umsetzung in Quantenlösungen überarbeiten. Aus diesem Grund könnte die Integration und der Einsatz von Quantencomputern aufwändig sein, und nicht alle Probleme können davon profitieren. In diesem frühen Phase benötigen die Beteiligten, z. B. Forscher und Unternehmen, daher Unterstützung bei der Entscheidung, ob sie Quantencomputer oder klassische Computer einsetzen wollen. Um dieses Problem anzugehen, stellen wir das Konzept eines Entscheidungsbaums vor, der den Benutzern bei diesem Prozess helfen soll. Der Entscheidungsbaum basiert auf Ergebnissen aus der Literatur, die mit dem Einsatz von Quantencomputern begonnen haben und deren Leistung mit klassischen Methoden vergleichen. Er besteht aus einer Reihe von Entscheidungen, die konsequent miteinander verknüpft sind, ausgehend von der Beschreibung der Anforderungen des Anwendungsfalls und deren Umwandlung in eine mathematische Formulierung, die auf Quantencomputern gelöst werden kann. Darüber hinaus wurde in diesem Projekt der inhaltliche Vergleich der verschiedenen Anwendungsfälle mit Hilfe von NLP-Methoden untersucht, um ähnliche Fälle identifizieren und eine Empfehlung auf der Grundlage ähnlicher Inhalte zu geben. Der Entscheidungsbaum muss allen Nutzern mit unterschiedlichem Hintergrund, einschließlich Forschern, Technikern, Anbietern und Entscheidungsträgern in der Wirtschaft, auf einfache Weise zugänglich sein. Um die praktische Durchführbarkeit unseres Konzepts zu prüfen, stellen wir QuCSplit als Framework vor, der den Benutzern bei ihren Entscheidungen hilft. Der Entscheidungsbaum-Ansatz kann ein zuverlässiges Modell für lineare Entscheidungen liefern, die sequentiell geordnet sind. Es ist jedoch immer noch schwierig, Entscheidungen zu erfassen, die komplexe Berechnungen und Beziehungen erfordern. Darüber hinaus ist die Methode des Textvergleichs noch auf der Suche nach einer zuverlässigen Evaluierungsstudie, um ihre Genauigkeit zu analysieren.

# Contents

# 1 Introduction

Quantum computing is a new emerging technology that can break many barriers computer science has faced, especially by solving problems requiring less time and memory than classical computers. The new technology based on quantum mechanics can provide an exponential speedup, which can overcome different classical algorithms in variant domains such as optimization, simulation, and cryptography [BBB+21]. For this reason, researchers and vendors such as IBM, Rigetti, and D-Wave are interested in the development of quantum computers nowadays and were able to introduce the first commercial quantum computers. These computers are accessible as a cloud service associated with many tools and services. These tools are built in a stack of layers architecture, where each layer aims to abstract from hardware and low-level details in order to enable easy access and development of software quantum applications. Furthermore, it allows involving more stakeholders and developers from different domains to apply variant use cases as quantum applications.

Based on the report [BHO+22], the development and deployment of quantum technology is still not a straightforward process. Although quantum computing promises to help companies address encountering challenges and overcome the traditional speed of classical computers, use cases are primarily experimental and conceptual at this early stage. Indeed, researchers are still arguing the most fundamental issues in the subject. That's why chief information officers and other executives waiting for quantum computing news must no longer be passive spectators and be involved in the development process. Instead, they should begin sketching their strategy for integrating and deploying quantum computing into their real-life applications, particularly in businesses such as pharmaceuticals that could take advances of early commercial quantum computing, as several firms expect to use a practical quantum device will by 2030.

The illustration of such a strategy to apply business cases has a critical restriction. Business leaders focus more on providing a high-level description of the problem statement and the requirements of each solution that can deal with it. Thanks to the present technology in classical computing, they do not have to care much more about low-level details which describe the technical level, the physical device executing the program, and the used compilers and tools to run it. In contrast, quantum computing proposes a new programming paradigm different from the classical one. It forms a setback in the

development progress since one has to investigate which parts of the use case can benefit from the quantum technology and can be programmed through it. This gap between the context-related requirements and the new technical specification opens a new research space to determine how we can outline the development process, from requirements to implementation specified by a set of decisions. In addition, one needs to consider the current limitations of quantum computers, such as the error-prone and capitates, i.e., the small number of provided qubits,– The basic unit in a quantum computer–, or the topology which determines how these qubits can be connected to each other because it influences the rules and decisions we make through the whole development. Thus, the necessary guidance on how to use quantum computing is still not complete, which presents a difficult challenge to resolve in any case.

The implementation of use-case requirements and translating them into quantum applications are defined on the knowledge base, which includes facts, algorithms, and rules starting from higher levels to the operations on the quantum hardware. It can describe minor technicalities, .i.e, the number of variables the quantum computer can process, as well as platform-independent descriptions of the use cases like the details of the Vehicle Routing problem and its variants. Hence, we must bind between all levels to construct an automated decision-making process. This project studies the following research problem:

***RQ: How can applications and researchers be supported in the quantum-classical decision process?***

In general, the decision-making process is based on theories as well as previous practical experience, which one should analyze as the first step of this process. This raises the question of how we can retrieve and derive decisions from experiments applied for different use cases on quantum and classical computers. In the first place, it needs to investigate how we can structure the knowledge in a database accessible to stakeholders interested in quantum applications, including vendors, researchers, and decision-makers. To inspect the user requirements precisely, we have to check how can an interactive questionnaire be generated from the knowledge base to determine the best decision for the given requirements. These requirements should be formulated in order to reclaim the proper decision on whether to use quantum or classical computers. An important frequent issue is how to support new given use cases not stored previously in the database.

To address these research questions, we introduce a decision tree approach that helps the stakeholders decide which quantum and classical computing technologies can solve their cases. In addition, we investigate the contribution of use cases' descriptions by providing decisions based on similar cases. To enable easier access and benefit of this

decision tree, we implement QuCSplit as a support framework that realizes a fully automated decision support system, including the main features: (i) Store and access the decisive facts derived regarding different use cases using a graph database, (ii) enable interactive access with user querying information about a specific use case, (iii) process requirements given from the user either through an interactive questionnaire or from a text description, and (iv) provide endpoints and integration with other systems, i.e., QuAntil project [22e].

## Outline

The thesis is structured as follows:

**Chapter 2 – Background and Fundamentals:** It explains briefly the quantum computing concepts and how it can be used in different domains. In addition, graph data based and decision rule concepts are described.

**Chapter 3 – Related Work:** An Overview of related work for this project. First, we review literature which use quantum methods and compare it with classical computing for a specific use case. Then, we report recommendation system concept that use the content of documents to provide similar results to a given query from the user.

**Chapter 4 – System Concept:** The contribution of this project is presented to solve the problem statement and the framework architecture justified with the design decisions.

**Chapter 5 – Implementation:** It demonstrates the implementation of the concepts and architecture decision mentioned in the previous chapter.

**Chapter 6 – Evaluation:** We define the test methodology of the features against the requirements as a proof of concept.

**Chapter 7 – Conclusion and Future Work:** A summary of the thesis's work and discuss the possible future work.

# 2 Background and Fundamentals

In this chapter, we explain the main fundamentals and necessary background knowledge for this project. Quantum computers are built using the concepts of quantum physics, which is different from classical computers. Thus, it is essential to explain how quantum computers work and their applications in different cases in Section 2.1. All facts and information retrieved from various experiments and reports must be kept in a database to enable an efficient and reasonable interpretation. Section 2.2 introduces the concept of graph databases, which are a type of non-tabular database. An important use case is visualising the decision tree, which is mentioned later in this section. To handle new use cases, contents can be compared to get similar known topics. The method is based on text matching, which aims to detect the important common words between texts. The concept is explained in section Section 2.3.

## 2.1 Quantum Computer

Quantum computing relies on the concepts of quantum mechanics, which describes the physical features at the atom scale. In quantum information, we consider the qubit as an atomic computation unit which can interact with other qubits, not as a stream of bits that classically encode a different kind of information. We explain the concepts of quantum physics briefly to clarify the structure and workflow of a quantum computer

### 2.1.1 Quantum mechanics

Objects at the atomic scale, rules describing their attributes and behaviours differ from classical physics. For example, energy and momentum can have only discrete values in quantum mechanics. Each object has a dual behaviour that can have the characteristics of both wave and object. That means that it always has an uncertain state, which implies that measuring at this scale is uncertain. In other words, trying to read a value for any physical attribute is unpredictable and not stable as in the classical way [GS18].

Describing the state of objects in quantum mechanics is different because we can not define the state at a specific time before measurement. As an example, a Bit (or a Qubit in the quantum case) can have the value "0" or "1", introducing a new type of state, which is superposition. All possible states can be represented through a linear combination, where each state has a fixed probability of being produced after the measurement [Hid21]. It enables storing all possible values, which represent configuration, solution, and state encoding, in one qubit in contrast to the classical bit, which can hold only one value. Thus, in a search problem with 100 possible keys, we need 100 qubits instead of $2^{100}$ classical bits because each qubit can be superpositioned of "0" and "1" states.

The act of measurement denotes detecting a specific property of the system, reading the current value, and one can perform another measurement. In quantum mechanics, The measurement transforms the qubits from their indeterminate state to a classical determinate state, which damages any superposition state [Hid21]. That's why it comes at the end of the quantum phase since it is irreversible. After performing all necessary steps, we can retrieve the information by measuring the qubits. However, since the measurement is uncertain, the quantum computer might generate the wrong or undesired solutions, although the computation steps are proven to deliver the correct solutions each time it is applied.

Entanglement describes a physical phenomenon that shows a strong correlation between the states of two objects, even if long distances separate them. When a part of the system is measured, this affects the result that will be retrieved by measuring the other entangled part. In quantum computing, entanglement improves the processing speed of quantum computers. As research [JL03] mentioned, quantum entanglement is necessary for a quantum algorithm to offer an exponential speed-up that overcomes the performance of classical computations as the main contribution in computer science. Another benefit is that fewer qubits are required to observe since we can interpret the state of the other entangled qubits.

### 2.1.2 Quantum Computer & Algorithms

The quantum computer is a device that uses quantum mechanics properties to perform computations per J.D. Hidary [Hid21]. The basic definitions in classical computers, such as bit, gate, and register, will be transformed to adapt the quantum mechanics concepts. A qubit differs from a classical bit in that it can hold the value 0 or 1 as states and a superposition of both states. Thus, it is represented as a vector of length$N$ – the number of all possible states – where each component of the vector holds the coefficient of each state. Furthermore, we can store qubits in a quantum register, where these qubits can be in one of the states or a superposition state. Then we can apply the computation steps,

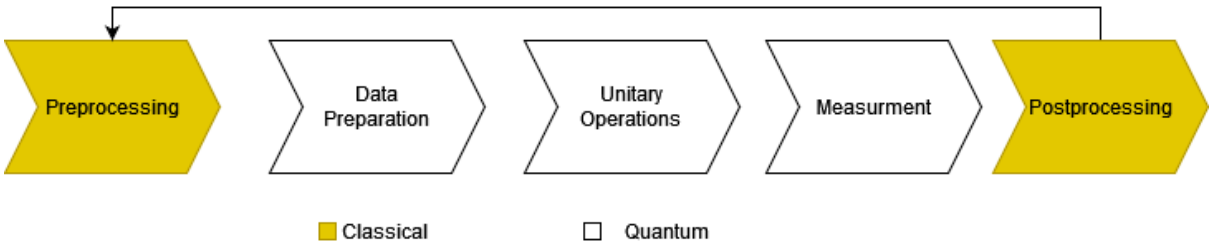


**Figure 2.1:** Quantum algorithm consists of five main steps [LBF+20]

which are unitary operations or represented as quantum gates, to those qubits to process the data. The unitary operations are the cornerstone of the quantum algorithm.

A quantum algorithm extends the definition of an algorithm by including extra steps required for data encoding, quantum state preparation, and post-processing of the results that come out from measurement. As shown in Figure 2.1, the quantum algorithm consists of five main steps. The first two steps are necessary to initialize the quantum state. This can usually be done by applying the Hadamard Operator on a classical bit to retrieve the superposition state. The primary processing is the stream of the unitary operations, completed by measuring the system's state and processing the output result. The quantum algorithm still depends on some classical operation related to the preprocessing of the data, which is originally encode it in classical way, and the post-processing of quantum results into the classical encoding [LBF+20]. To solve the measurement uncertainty, one needs to enhance the coefficient of the correct solution and reduce the other solutions. After finishing the computation steps, we increase the coefficient of the correct output and minimize the other ones without performing any measurements. This can be done by applying specific unitary operations as well. Finally, to get the highest probability of a correct answer, we do this process iteratively until the coefficient (amplitude) can not be increased anymore, known as Souffle Effect [Bra97].

### 2.1.3 Applications of Quantum Computing

Quantum computing can unleash significant business values across industries, ranging from cryptography, machine learning, simulation, and optimization. McKinsey's report [BHO+22] defined all use cases under four types: (i) quantum simulation, (ii) quantum linear algebra for AI and machine learning, (iii) quantum optimization and search, and (iv) quantum factorization. Following, we explain each type briefly:

**Quantum simulation:** The modelling of quantum-mechanical systems or processes such as molecules, chemical reactions, or electrons in solids is known as quantum simulation.

Such problems are tackled roughly because conventional computers cannot correctly mimic quantum systems with more than a few dozen particles. Quantum computers, on the other hand, are innately suited for such complex quantum simulations since they are based on quantum physical systems, providing an exponential speedup over traditional computers. Quantum-simulation applications also include lead discovery and catalyst improvement.

**Quantum for AI:** In some applications, quantum algorithms can replace critical steps in traditional machine-learning pipelines with a demonstrated quantum speedup to reduce training time. In some applications, such as quantum neural networks, the whole learning technique is moved into the quantum domain. It outperforms traditional algorithms but demands hardware requirements such as memory and greater fault tolerance, as well as a well-defined mathematical formulation of the given problem. Quantum technology advances natural language processing by proper extracting meaning from complicated words and forming outputs. Furthermore, it is beneficial for automating difficult jobs such as financial recommendations. Quantum AI and machine learning may also be used in other sectors, including pharmaceuticals, automotive, and finance, to perform different tasks such as autonomous driving, automated trading, and predictive maintenance.

**Quantum optimization:** The quantum approach might tackle previously intractable problems by finding better answers in the same amount of time; for example, quantum algorithms for discrete optimization often yield a quadratic speedup over traditional computing. Experts predict it will facilitate real-time optimization by reducing computation durations from hours to seconds. There are applications for quantum optimization in practically every field, including generative design, traffic management, and portfolio optimization. In addition, quantum search provides a quadratic speedup to identify specific entries in an unstructured database. Similar techniques are used to improve traditional Monte Carlo simulations, widely used in quantitative finance for essential tasks such as derivative pricing. Quantum search can reduce computation time from days to hours, allowing for faster trading choices.

**Quantum factorization:** The most well-known use of quantum computing is quantum factorization. Shor's approach for computing prime factors of large numbers was one of the first methods. It has been mathematically proved that it provides an exponential speedup over the best conventional algorithm available at the time. Efficient quantum factorization is most easily applied to breaking Rivest–Shamir–Adleman (RSA) encryption, which serves as the foundation for the majority of today's secure data-transfer methods. We do not expect a significant cybersecurity problem because post-quantum encryption methods, alternatives to RSA encryption, are currently being developed. That's why organizations should improve their encryption standards in the coming years.

Automotive Industry is one of the most benefited domains of quantum computing as mentioned in the report [BHO+22]. the acceleration of finite element simulations used by (OEM)s and suppliers to predict vehicle mechanical stability, aerodynamic attributes, thermodynamic behavior, and noise, vibration, and harshness. Increasing the speed and reality of these simulations may provide value by lowering the cost of prototyping and testing, as well as producing better, higher-performance designs at a cheaper cost. Quantum computing will also definitely improve autonomous driving. Faster machine-learning models can boost R&D cycles, and synthetic data created by quantum computing can minimize the cost of gathering and labeling data and improve vehicle performance in unusual situations, which come with little real-world training data. Another use case which we will discuss in chapter 3 for quantum computing is prototyping and testing, which presently accounts for 20 to 30 percent of the overall $100 billion R&D cost of a new vehicle, including hardware components and vehicle assembly and testing, according to the study. The speed and capacity of quantum computing to accomplish complicated tasks that traditional computers cannot handle may allow for more virtual testing in the future, reducing the number of test vehicles necessary. Prototyping and testing costs have already been reduced by half thanks to high-performance computing. Quantum computing is likely to enable additional savings by minimizing calculation times, allowing for more tests, and improving accuracy.

However, another report [Ond20] expects that quantum computing is unlikely to replace present high-performance computing (HPC), and the earliest attempts at value generation will not rely completely on QC devices that address whole problems. Instead, researchers intend to use the quantum technology in cases that will primarily apply hybrid approaches (both classic and quantum computing). In the beginning, the quantum function will rapidly produce an approximate solution to an optimization issue. Then, A traditional HPC will improve this solution using a smaller set of factors. This allows programmers to use the current stage of quantum computing to run HPCs more effectively.

Furthermore, it states that the motivation behind finding alternative integration types instead of complete dependency of quantum computing is its uncertain future [Ond20]. Businesses must evaluate their full range of use cases for the technology over distinct time horizons. While most quantum computing will not be commercially available for at least 10 years, automotive vendors should explore for options in the short term (the next one to two years), in which they can use quantum computing. They may start by identifying potential cases in the value chain, developing research partnerships and intellectual property, assembling a small staff, and setting practices. Large IT corporations, academic institutions, government laboratories, and start-ups staffed by quantum-software engineers and other specialists are all potential partners. In the midterm, people should focus on application development and focused capabilities. During the process, they should identify front-runners, expand teams, and launch the

first prototypes. For a longer term, over the next 10 years, organizations should develop a technological edge via quantum computing, establish a competitive advantage in certain sectors, and begin to improve their core skills.

## 2.2 Graph-based Databases & Decision Tree

This section describes the graph database, which is a NoSQL database. Instead of saving the data in tables, it represents the items as a graph. Through graph database, decision rules can be represented as a tree structure which illustrates the consequences and outcomes of each decision, as explained later.

### 2.2.1 Graph Database

Facts gathered from different experiments represent the knowledge from which we can derive recommendations and decisions for users' use cases. This knowledge can be visualized through a graph database. A knowledge model, a set of interconnected descriptions of ideas, entities, relations, and events, is the core of knowledge graphs. Knowledge graphs can contextualize data by connecting semantic information, providing a foundation for data integration, unification, analytics, and sharing. Knowledge graphs have the three characteristics of several data management paradigms: (i) Database because it includes data that can be retrieved via user queries. (ii) Graph, since it can be analyzed as any other network data structure, and (iii) Knowledge base that includes formal semantics, which can be used to understand the data, analyze it and infer new facts [22g].

A graph is a network-like structure consisting of nodes interconnected by edges. As an example, OrientDB's graph model [20a] is represented by the concept of a property graph, including two main types of entities : (i) Node, which represents a state, snapshot, or a set of information that can be connected with other nodes, and Edge that connects two nodes. Hence, it is an entity with two prime attributes describing the incoming and outcoming nodes in addition to its unique identifier.

A Node or edge entity can have multiple subtypes. Each type extends the base graph model to include an additional scheme. This serves to build up a formal semantics of the model, which can provide a shared understanding and interpretation of the graph in a precise manner. Briefly, we model the questions or the current state of knowledge through nodes, and the new answers jump to further states until we can decide which methods of quantum or classic we can use, as shown in Figure 2.2. We explain it in more details in the later sections

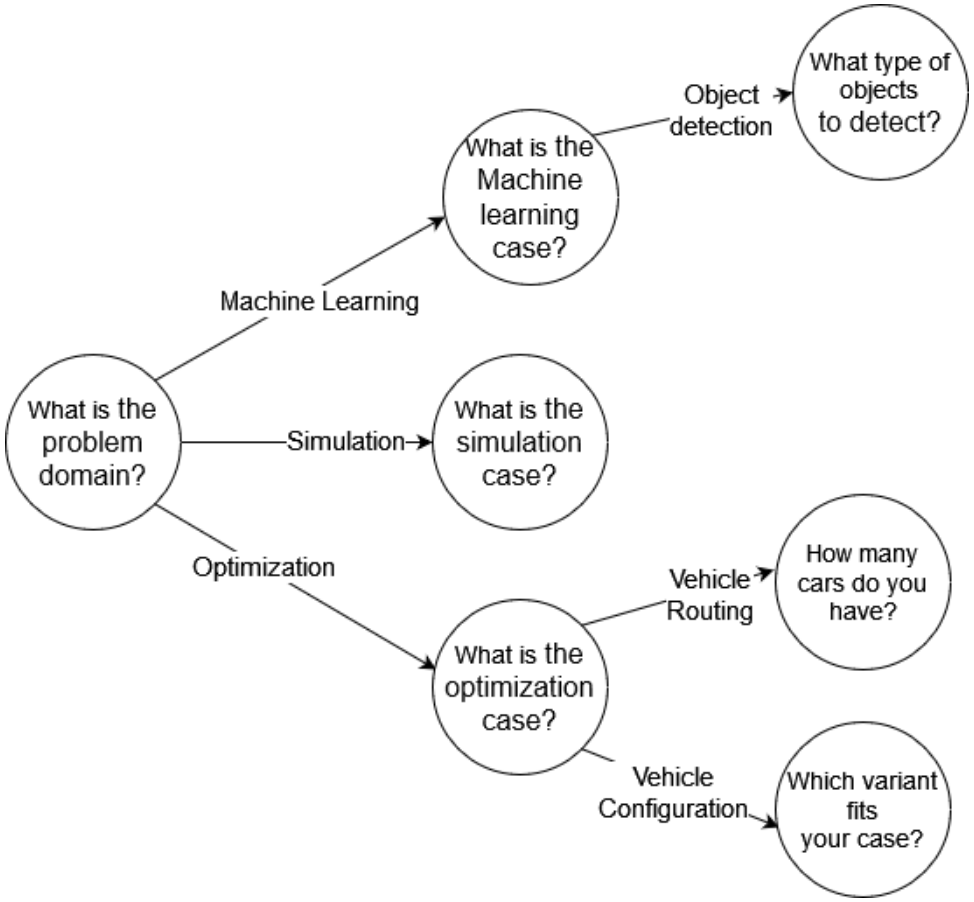


**Figure 2.2:** An example of a graph database

### 2.2.2 Decision Tree

A decision tree [KJS18] is a tree model that includes a set of decisions connected with possible consequences such as events, risks, and costs. It helps to build up a strategy in decision analysis to let the users reach an end goal. Each node represents an assessment of specific information, and each outcoming edge is associated with the possible outcome of this assessment. Hence, a decision tree can be considered as a property graph in which the internal nodes are part of the final decision argumentation. Compared to our case, the questions aim to retrieve information from the user to get a sub-decision on the current node. Then, it lets to proceed to the next node for another decision. These sequences of decisions can help to reach the final goal by selecting the best technology.

The decision tree can be linearized [Qui87]. Thus, all the statements along the path can form a conjunction which is satisfied when the outcome is the final decision node. The linearization forms the fundamentals for mapping between rules described in simple text

into a graph model. In this way, it eases turning regulations and statements extracted from publications into questions represented by nodes in the graph database.

However, the decision tree as an approach in decision analysis is entirely deterministic, which means it can not detect similar decisions or the outgoing edges based on their content but only based on the tree structure. That's why we introduce the text-matching topic in the next section.

## 2.3 Text Matching

Text comparison is a crucial application in natural language processing and information retrieval. We aim to compare the texts to retrieve similar ones related to a given text. It is considered a critical task because texts can be formulated in different forms with various choices of words. In addition, we are not always concerned about a complete matching between texts but with related ones clarifying the similarity ratio.

The vector space model is fundamental to many information retrieval operations, including query scoring, document classification, and document clustering. In this model, we represent each document as a vector, where each component holds a piece of information about this document. For example, it can be the occurrence of words found in this document, tf-idf or other weights. Turning a set of documents into a measurable space can allow us to compute the similarity using a scale like the Euclidean distance between two points, which allows defining the relative positions of these points. However, calculating the euclidean distance between vectors as a similarity measure has a critical drawback: two documents with similar content might have a vast distance because they differ only in length. Thus, the distributions of terms may be identical in the two documents, but the absolute term frequencies of one may be far larger than another [Man08]. That's why we use the cosine similarity measure, which defines the angle between two vectors. In the case of information retrieval and content comparison, we get a value between 0 (not similar at all) and 1 (identical). Negative values are ignored here because we do not consider the term's meaning and its opposite. Hence, the two documents have similar contents when their vectors' angle is smaller, as in Figure 2.3.

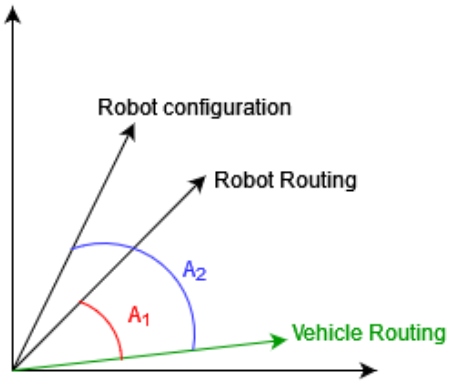


**Figure 2.3:** Robotic Routing use case description is similar to Vehicle routing than Robot Configuration

The text matching in this project is used to compare the use cases' descriptions. Each use case includes keywords that refer to other related use cases. Hence, there is a significant likelihood between those cases if they describe problems in the same domain. Moreover, we see later in chapter 3 that when cases are modelled using the same class or formulation, they are more likely to get similar results. Therefore the occurrence of specific keywords, such as the problem class or used algorithm, has a particular impact on the similarity between cases. We use the concept of text matching and comparison in case of new given problems and searching for some similar ones in our database.

# 3 Related Work

In this chapter, we review the relevant research our work is based on. First, we discuss the researchers' works in section 3.1, trying to improve and integrate the quantum technology and applying them in different industrial use cases. The publications propose the definition of a reference case in which we find a description of the problem, the method used to solve it and the experimental results. These cases serve as a reference for other ones in the same domains. Second, section 3.2 reviews the latest related work on the development of a recommendation system that could aid the decision process of the user by getting content-based similar entries to the given query.

## 3.1 Industrial Use Cases: Quantum vs Classic

The impact of the integration of quantum computing is the central focus of the latest research. It has benefits in different industrial sectors, especially in automotive, since quantum approaches promise to overcome the classical ones. That's why it is crucial to identify high-level problems which can seek a potential use of quantum approaches to guide the development of this technology and seek a way to bind those cases with the use of quantum computing. BMW Group [LKP21] investigated both sides, high-level requirements and technical details of quantum computing concurrently. In this way, the improvement of quantum methods can inspire to solve new cases and the defined industrial problems can bundle the development of the quantum ecosystem. Their investigation concludes with dividing each use case description into four layers, starting from high-level domains to the quantum system description. In QUTAC [BBB+21], authors look deeper into the term of reference use cases. To ensure consistency in cross-industry discussions, they put definitions for each of the main terms of a reference case. We mention the definitions and clarify the meaning of each term as part of a reference use case.

**Problem domain:** The area for all problems which are characterized by the same goal, using similar computational methods to solve them.

**Problem Class:** A set of all cases which share the same abstract mathematical and complexity level.

**Model:** The model can represent the problem class as a computational problem where business or context actions are mapped to arithmetic or logical operation (e.g. The existence of a specific feature can be represented as a variable in the defined function)

**Algorithm:** The algorithm consists of finite steps of quantum computations to solve a specific problem.

The connection between the terms is shown in Figure 3.1, each layer describes a specific level of information related to the use case, starting with high-level information till we can reach the best fitting algorithm that can solve it. The connection describes the mapping between two types of information. A use case in the optimization domain describes a set of components of the problem, which can be mapped to one of the problem classes, such as Travelling Salesman Problem (TSP) and can go further to the next layers.

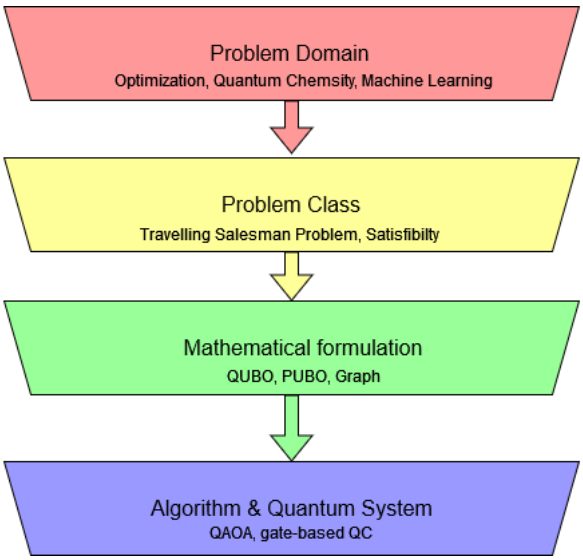


**Figure 3.1:** Four layers of each industrial use case [LKP21]

QUTAC [BBB+21] points out the generic property of a use case to describe all possible and similar cases because it would be the source to evaluate, improve, and develop the entire quantum ecosystem. Furthermore, it suggests that each use case should satisfy specific requirements to be considered a reference problem. In this way, these requirements can guide the research more to pick out representative cases. Thus, in each case, the following perspective should be considered: (i) Business impact, which denotes the improvement of using the quantum technology in the use case on the services, products, or processes which include it, (ii) generality, that describes how the solution of this use case can be adapted to solve similar use cases in the same domain, (iii) access

to guarantee that issues are clearly defined in a single vocabulary, adequately abstracted, codified, and intelligible, and (iv) technical feasibility to ensure that the use case can be precisely formalized, evaluated based on both present and future technology, and well-defined metrics can be developed.

In summary, The usage of reference use cases provides a template for any case, including the description of the problem, its assessment at the business level, the problem class and formulation, and the used algorithm. It can support further investigation and add new tools/methods/classes to the system. The four layers architecture facilitates the collaboration between all fields. The separation of application context from further mathematical details lets a broad of researchers and industry communities be involved by improving the quantum ecosystem. In addition, the defined problem class can guide researchers to develop new hardware and software for quantum computing to solve this specific class of problems. This serves as a basis for the investigation and the analytic evaluation of quantum approaches. We followed the structure of the reference use case to construct our knowledge database. The user starts by giving facts about the use case, which is connected to the problem class layer, depending on the previous experiment to solve this problem on quantum computers. Moreover, it lets us gather all reference cases in a structured table, where each column describes a layer of the four main ones, which extends the table mentioned [LKP21]. We collected all mentioned use cases and put them in a table[1]. However, these cases have not yet been solved through the suggested quantum algorithms.

One open challenge is to define application-centric metrics that help users from different fields to evaluate and decide how the quantum computing might be useful. The current benchmarks concentrate only on low-level details such as hardware performance and target providers, defining important metrics like quantum volume [CBS+19] or speed [WPJ+21] to assess the used technology. However, we can not directly bind these metrics to a real-world application performance because it does not investigate the complete end-to-end application performance. For this reason, the assessment process should go through all levels of a use case. QUARK [FRH+22] is a framework to advance the analysis and creation of application benchmarks for quantum computing. It makes it easier to plan, carry out, conduct, and bind application benchmarks through different levels. Due to the wide variety of applications, it is crucial to offer a flexible framework that inspects system performance on application-level quality metrics (for example, the path length for TSP applications), integrating the gap between current benchmarks and the applications. The architecture of this framework in Figure 3.2 is similar to the template of the defined reference use case. It starts with a description from the user about the intended application, then provides the possible benchmarks on the following

[1] https://github.com/mohamad-altaweel/QUCSplit

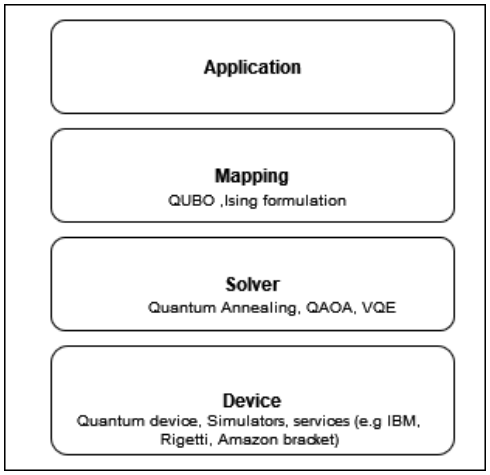


**Figure 3.2:** QUARK benchmarking manager architecture [FRH+22]

low levels until it reaches the final target device. The provided example [2] clarifies the communication with the user through single-choice questions. Hence, the occurrence of a question is dependent on the previous one and the given answer, which determines the following possible options. In this project, we are motivated by QUARK architecture that organize the series of questions to the user but with two main differences:

- QUARK considers guiding the end-user to apply a specific problem until it reaches the use of a device without providing any clues for the user which option could deliver better results from different aspects. Our goal is to recommend the user between the available options based on previous experiments on the devices. The final decision should be the optimal decision for the user to apply.
- QUARK does not start from the domain. As shown in Figure 3.2, the application top layer here targets the problem class. Thus, it could limit the public who can benefit from or use the framework because it needs prior knowledge of mapping between the real use case and its problem class. Therefore, we focus more on the information regarding the context of each use case and can ignore the details on lower levels and set it to be optional information which the user can specify.

To put it briefly, all recent researches agree that it is necessary to build a framework based on a referral scheme to enable easier access and integration of quantum computing into different use cases. It should help the user by translating their requirements derived from real-life cases into solutions provided by quantum computers. Otherwise, it would be challenging to apply each case from each sector without knowledge exchange from

[2] https://github.com/BMW-Group-Quantum/QUARK

other stakeholders and raise many obstacles that block this technology's integration process.

## 3.2 Content-based recommendation system

A recommendation system assists users in making decisions when they do not have enough personal experience with all the other alternatives [RV97]. These systems are used in various fields, including entertainment options like books, music, films, healthcare, smart homes, and mobility [BP14] [ASMD19]. Recommendation systems have shown effective success in suggesting and decision-making processes; nevertheless, when little information is provided, their performance drops substantially, whether given from a manual user or retrieved from different data sources. In our project, we face the issue when the use case is not stored in our knowledge database or when the user needs some hint and no further answers can be provided to step forward in the graph database.

Bergamaschi and Po [BP14] proposed a plot-based recommendation system for movies. It is only based on information retrieved from the plots of the movies and formed as a list of keywords in the Vector Space Model. The similarity approach between the queried movie and possible candidates has four main steps:

1. Movie Vectorization: It starts by extracting keywords, removing stop words, and applying lemmatization techniques.
2. Weights Computation: weights are coefficients of the word index and denote the occurrences of keywords in the plots. Later they are modified by using the tf-idf technique.
3. Matrix Reduction by using Topic Models: Each document could contain hundreds of keywords, which leads to increasing computation time. That's why the matrix is reduced to a lower dimensional space using the Topic Models methods LDA and LSA.
4. Movie Similarity Computation: similarity between two plots is computed by considering their topics from the previous step using cosine similarity.

In our project, we follow the main four steps mentioned in [BP14]. However, we modified the procedures done in each step in order to fulfill the requirements of the project, as explained in the next section.

## Keyword extraction

Keyword extraction is a critical task in text matching and documents comparison. It is a crucial step in order to bring all documents in one space with the same size and no need to handle irrelevant words in each document as the first step in [BP14]. Synthetic keywords approaches depend on pre-defined POS patterns to get a list of keywords. Compared with generative analytic approaches, it can better satisfy the well-formedness property of the extracted keywords. Keywords extraction can be considered either as (i) a classification problem in case we have a training set of words labelled as keywords or non-keyword or (ii) as a ranking problem, where each word has a score gained using different kinds of techniques. Unsupervised methods can be divided into three main types: (i) statistical, which uses statistical features in order to extract the most significant words and phrases from a given text .i.e YAKE!, tf-idf, RAKE, (ii) entropy-based, which assumes 'keyness' property is represented in the spatial distribution of the occurrences of words, (iii) graph-based methods which concern about the connectivity between the words and to inspect the 'centrality' of a keyword, which denotes the representativity of the keywords .i.e TextRank, SingleRank, ExpandRank, TopicRank, PageRank, PositionRank [FNAD20].

To sum up, the decision-making process could be based on different resources such as information retrieval, features detection, or fixed predicate logical rules. Recommendation systems can use text matching and keywords extraction to compare the content of documents and inspect the similarity between them based on some features related to the extracted keywords. We use text matching to enable users to get hints or similar results when the system can not find a precise answer to the input given, as we will see in the next section.

# 4 System Concept

This chapter describes the solution concept to solve the problem of this project. First, Section 4.1 defines the requirements that indicate the features necessary in our framework. Then, the method in favour of solving the problem statement and fulfilling the mentioned requirements is described as a systematic process in Section 4.2. In addition, we explicitly state our contribution based on the literature reviewed. In Section 4.3, the system architecture and the process steps which shape the final solution of this project are introduced. Finally, use cases comparison based on descriptions in Section 4.4 is an additional approach to support the graph model by solving unknown cases issue mentioned later.

## 4.1 System Requirements

The scope of this work is to assist users from different domains in determining when to use the quantum computing in a given case through a decision tree concept. It is important to enable easy access without needing of any prior knowledge. In addition, it can provide integration opportunities with other tools or platforms in order to build a common platform that offers different ways to use quantum technology. To address this issue, we list the requirements in this project's scope that should be included in the final solution and realize the contribution of it.

**Functional Requirement**

**R1.** Construct a knowledge database which includes all facts, algorithms, and necessary information to decide which art of computing to use.

**R2.** Implement a service that receive user requests and process the given information depending on the knowledge base.

**R3.** Define a user interface component, which can interact with user to get different type of inputs.

**Non-Functional requirement**

**NR1.** The framework should provide integration points to exchange information and outputs with other tools, specifically as part of QuAntiL project [22e].

**NR2.** The framework should be extendible to include further mechanism to derive decisions regarding splitting uses between quantum and classical.

## 4.2 Method & Contribution

Because of (i) the lack of experiments comparing quantum and classical computing on real use cases and (ii) the different methodology and metrics used, it is difficult to build a general scheme for the knowledge base, which can guarantee the adoption of future cases. Therefore, we start from a reference use case, extract the statements and decisive facts which impact on the results according to the report, rewrite them as questions to the user, and define the database structure based on it. Figure 4.1 shows that this process is done iteratively. Each time we grab the facts and results from a new experience and check how the database structure can be adjusted to be more generic. This means that the knowledge base follows a prototyping process, where each experiment enhances its final shape.

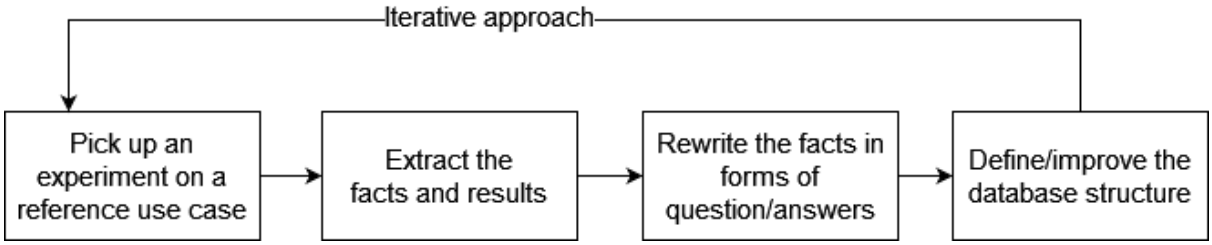


**Figure 4.1:** The work construction is an iterative process

We selected two reference use cases: Vehicle Configuration Optimization and Vehicle Routing Problem (VRP), which are noted in [BBB+21] as important applications in the industry field. In addition, comparative studies between the performance of quantum and classical computing to solve these problems are already published.

### 4.2.1 Vehicle Configuration Optimization

Optimization of pre-production vehicle configurations is one of the challenges in the automotive industry. Given a list of features and types of cars, each has a subset of the features; the goal is to find the minimum number of cars that include all the features

and obey the configuration rules. This use case plays a fundamental role in testing all features from the different types of cars [GKS22].

Despite the theory that states that quantum algorithms should solve many problems with better performance, results from experiments disagree with this statement. Glos et al. [GKS22] compare the CQM solver on D-Wave [22b] with classical solvers CBC [FL05] and Gurobi [Gur21]. It shows that classical solvers provide solutions faster. However, the CQM Solver can not deal with more than 5000 variables. By increasing the time limit to 10 seconds, the CQM solver produce a valid solution at each iteration. Real-world problems, which could not be solvable for classical solvers, often need even more variables with efficient timing. A related and more general problem is product configuration and reconfiguration, where the problem's scope is not restricted to vehicles, and one may consider any product, such as computer parts. Authors have developed a model that can be extended for such problems and evoke potential use cases for D-Wave CQM Solver. QUARK [FRH+22] presented the problem as a reference use case and used three benchmarks to compare classical and quantum methods: **V** Validity which denotes whether a solution found by the solver is valid, **TTS** which is the end-to-end time required to obtain a solution, and **Q** quantifies the quality of the solution. The results show that considering TTS the quantum annealing outperforms the classical solver starting from n = 90 number of given variables. Otherwise, classical methods still have better performance. From the previous results, we can generate the following prototype as a questionnaire scenario:

1. Q1.What is your problem domain?
   O. Optimization O. Simulation O. Machine Learning
2. Q2.What is your problem industry field?
   O. Automotive O. Logistic O. Materials & Design
3. Q3.What is your application Scenario?
   O. Vehicle Configuration O. System Verification O. Route Optimization
4. Q4.Which Variant of Vehicle Configuration does fit your case?
   O. Vehicle options optimizations O. Crash relevant component layout
5. Q5.What is the decisive factor in your context use case?
   O. Time-to-solve O. Validation O. Quality
6. Q6.(User cares about time) Please provide the number of features you have.
   O. $\leq 90 \implies$ Classical O. $> 90 \implies$ Quantum Annealing

To sum up, the simplest form of the question is the single choice question and the length of the path (number of questions) to answer might differentiate depending on the given answers, such as the Q5 in our example (Tree is not balanced). It is also important to retrieve the decisive factors (benchmarks) then we can continue asking more about the

problem description, as in Q6 we asked about time then we asked about number of given features after we know that time is the desired benchmark.

### 4.2.2 Vehicle Routing Problem

The Vehicle routing problem is a generalized formulation of TSP. We consider a set of vehicles; each must visit a group of cities once with the lowest possible cost, where each vehicle might have different travelling costs. Further constraints are applied to this problem. E.g. Each city has a demand for goods which vehicles should deliver, and each one has a fixed capacity, like the Knapsack problem.

The current resources and computational capability of quantum computers still make it hard for the quantum computer to overcome the classical methods. Azad et al. [ABA+22] solve the basic version of VRP using Quantum Approximation Optimization Algorithm (QAOA) and concluded that the classic solver still performs better. At the same time, other researches aimed to solve the constrained types of VRP, which include further conditions. Irie et al. [IWT+19] try to solve the Capacitated Vehicle Routing Problem (CVRP) version of this problem, in which each city has a particular demand that needs to be fulfilled at the end of the solution. In addition, it saves the state of each city, which records the arrival state and departure state. This introduces the time window by adding a new constraint to each city that could have specific available hours to receive demands. The results from execution on D-Wave 2000Q, show that the formulated problem works in small-size QUBO systems, which include less than 90 logical qubits. It allows using a problem instance with 6 to 7 customers and can be directly deployed in the current D-Wave machines. However, A real case scenario requires more than 2000 logical qubits, which represents more than 30 customers for 20-minutes scheduling of a half day. Another version is the Heterogeneous Vehicle Routing Problem (HVRP) which Fitzek et al. solved using QAOA [FGL+21]. The report summarizes that the classical approach still outperforms the quantum method. Another quantum approach Bennett et al. used [BMMW21] is the Quantum Walk-based Optimisation Algorithm (QWOA). The simulated experiment shows that the QWOA is able to produce near-optimal solutions for a randomly generated problem with eight locations.

We found out that the formulation and description of the problem have an impact on the final decision. For example, different variations have been conducted, showing different results. This infers new types of questions related to the problem description as already been proposed. For example, In VRP, if we consider the demand of each city, then ask about how many customers we have (in case of > 6 customers, quantum is better to use) [IWT+19]. Otherwise, The experiment on the basic version recommends the classic approach [ABA+22].

### 4.2.3 Open Issues of Different Experiments

Most works of literature consider some benchmarks regarding using the quantum approach as in the previous case. Each paper has its own designed experiment, leading to some problems. As part of this work contribution, we present these issues and how to handle them to be part of the knowledge a user can use to retrieve a reasonable decision.

**The heterogeneous experiments:** Each experiment might consider different factors or instances, making it difficult to put them together in one database to compare between each approach. For example, the Quantum Annealing (QA) approach could solve the problem of CVRP [IWT+19], which is harder than the basic one solved in previous work, but the results are different. Logically speaking, if QA could solve problems with six cities/customers of CVRP, it should also solve the VRP with the same number of cities, but this was not investigated. The quantum device is concerned only about the number of qubits and the way they are connected to each other, regardless of their representative context of the given problem. When every two variables are multiplied in the mathematical formulation, their qubits are coupled. Quantum devices follow a fixed architecture which defines the connections between qubits, which means some problems can not be solved regardless of their size or complexity [22f]. Hence, we can not transfer the complexity levels of problem variants into the quantum space without any modification. In this case, a CVRP instance must not always be harder than a VRP instance for a quantum computer to solve. One must check how each constraint added to the problem can increase the number of qubits required and their connections with each other. That's why we consider the problem variant a decisive statement that changes the possible algorithm or formulation used since we restrict to the experiment that solves this variant of the problem.

**The instances of the problem used in each experiment:** QWOA delivered a better solution than classical ones, but with instances with eight locations [BMMW21]. It raises the following issues: What if the user has more than eight locations? What if the graph has another pattern (planar graph, full connected, or tree)? We can not infer such information not investigated in the experiment. Experiments observe the performance of classical and quantum computers trying to increase the variables related to the problem like the number of customers, the available number of features, or the remaining number of required tests...etc. Each variable has a boundary point where quantum can overcome classical performance and vice versa. In this case, the boundary point must be detected and check if the given case from the user is above or lower than this boundary, .i.e number of customers given is less than five, and it is experimentally proved that the quantum computing is efficient to solve till seven customers.

**The used benchmark to compare:** QWOA has used the solution quality only [BMMW21]. What might it be if the user wants the approach which has a better time? When an experiment does not report one of the metrics, it represents a missing knowledge issue. In this case, the decisive point which method to use would be the desired metric or benchmark from the user, but not the decision which method performs better. That's why, the user should define which benchmark he is interested in to compare between classical and quantum approach

In summary, these open issues convert the need for a tool to give recommendations into motivation for a unified methodology in further experiments in the future. Moreover, they can help to shape them depending on a set of standards that each should have to be documented as a new finding for a specific case.

## 4.3 System Design & Architecture

Both requirements and issues mentioned previously have the major impact on the framework architecture. Therefore, it should reach this project's major task and enable the building a reliable software system. In this section, we introduce the overall architecture, explaining each part of it and augmenting the design decisions we made to solve each issue.

### 4.3.1 System Architecture

The functional requirements present the three main functions that the framework should have. Therefore, we split into three main components, each is responsible for one of the required functions, following the Three-Tire architecture. This architecture enables each component–or layer– to be run and developed separately without significantly impacting on the other components [Fow12]. In addition, we profit from it in the following points on the technical level: (i) Easier development that states any update or change in one layer has a minimal change in the other layers, which guarantees an easier integration between the layers. E.g. Adding new information in the database will not change the user interface layer. Moreover, the database concerns only with saving the data regardless of the decision process, which is done by the application layer. (ii) Reliability improvement implies that failures in the application layer will not lead to an outage in the database layer. This improves the integration of components and new features easily when each layer is concerned about executing its expected functionality.

However, layers still depend on each other to do their functions properly. Figure 4.2 defines the dependencies between layers. The application layer should connect to the

database to retrieve the statements and facts related to the user's request. Otherwise, it can not process the data and proceed to the final decision. Similarly, the user interface needs to get the information from the application layer to display it. This defines the dependencies as a one-way line from the database to the interface, which means that to run layer X, we need to run the next right one, which is implicit in running the following consecutive layers.


**Figure 4.2:** The dependency between layers is a one way direction

To conclude, the three-layers architecture is the primary pattern of the framework. Each layer is responsible to provide a set of endpoints to the next layer, trying to reduce the number of assumptions from the other layers about its internal functionality and logic.

### 4.3.2 Database Layer

The database layer is responsible for saving the knowledge gathered from different experiments, which can aid in deciding which method to use between quantum or classical. It should maintain the data mapped from textual statements in a form that can enable us to interpret and compose them together reasonably. That's why we consider database construction as the critical contribution of this work because it addresses the central part of the research question of this work, mainly using the prior knowledge and experiments used in quantum computing to help in the decision-making process of splitting between quantum and classical use cases.

We introduce the database type used, define the scheme of the knowledge base, and explain how we model the facts as a graph of nodes whose connection represents logical interpretations. The graph database has issues in case of user missing inputs. We review the main problems and introduce our solutions to them.

#### 4.3.2.1 Database Type

The type of database defines how is the data saved in the database. We compare between the different types to decide which one fits our case.

| — | Relational Database | Non-relational database (NoSQL) |
|---|---|---|
| definition | Data is stored in one or more tables of rows and coloumns | Store data as individual entities without any pre-defined rule or shape |
| implementation | Each question is table and answers are its entites. Each entitiy connects to the next question | Keep the original data format and define a relationship rule i.e. graph |
| Pros | Efficient search | Preserve the original structure |
| Cons | Rules restrict the model | Inefficient performance |

**Table 4.1:** Main two types of databases

The relational database (rule-based) seeks to provide the data efficiently. However, The rules between the tables can restrict the base's model and make it harder to handle compared to NoSQL models in many ways: (i) One can not differentiate between tables if we have a special type of questions or statements. Furthermore, all tables are abstracted, so we can not easily predict or expect the type of answers while we typecast entities in NoSQL. (ii) Two cases might end up with the same question, especially about the problem class. So we duplicate the table to split between the two questions, which results in many duplicates for the same question because we can not link the previous answer to the table. With NoSQL, we can define an instance related to the previous question for such questions so as not to lose the whole path. (iii) Relationships are entities with special types or additional information, not only as a connector between entities. That's why the relational model could not fit our case as the Non-relational model.

NoSQL databases have also many types: (i) document-based, in which each object is saved as an object and could have a special format like JSON, (ii) Key-value databases are a simpler type of relational database where the table includes two main columns; a key and the content of the item, and (iii) Graph databases store data in form of nodes and edges. Graph and document database types can provide a good model to question/answers entities or connect a set of facts with each other. However, we chose to model the knowledge to reduce the amount of read and write operations from files, especially that objects here does not include an internal complex structure. Each object represents a statement which is a textual description and has a unique name for identification connected with other objects, as we see in the next part.

#### 4.3.2.2 Database Schema

The database schema defines how the data is organized. It is a blueprint that includes logical constraints and the relationships between these entities. Data modelling starts with the creation of the schema in order to get the data instances in the database. In contrast to relation databases, graph database allows defining of a metamodel that represents the semantics of the data.

We use the nodes to represent the current knowledge state to make a proper decision. When the user passes an answer, it navigates in the graph to a new state depending on the given answer. Thus, the answer or new information connects the nodes to reach a final knowledge state which informs which method to use. In this case, the graph is a tree type where one starts from the root node that represents no knowledge and passes through the nodes with a set of information to end up either in quantum or classic computing. One crucial issue is the sequence of facts or answers given in the graph to enable an easier and ordered exploration. Some facts might depend on previous answers, not only the current state. As an example, we can use different problem classes to model an optimization problem, but we need to know only the options which have been previously used and tried. Therefore, knowing which use case is asked is necessary to get only options tried with it. Since we are motivated to link the highly abstracted requirements that come from applicable use cases with the new technologies offered by quantum computers and referring to the primary goal of this work, we follow the template set from [LKP21] [BBB+21] to define the reference use case. Thus, we start querying information about the content of the use case, including its domain and descriptions. Then, this information could be mapped to the proper problem class according to the published formulation and quantum technology experiments. Figure 4.3 shows an example of the knowledge tree. It starts from the root node and enhances the knowledge state with general questions about the problem domain. When the use case is detected, more specific information can be gathered. In this case, some options will be automatically emitted from the available options as in use cases of the automotive industry, which does not include any case related to the cryptography domain as in Figure 4.3. That's why one does not have to ask for information related to this domain. The implementation of this order might vary depending on different use cases in response to issues mentioned previously, such as the different benchmarking and used instances. Some variants might be using the same problem class. However, the decision on the quantum performance will depend in the end on the number of variables included in the problem, as in the Vehicle configuration use case.

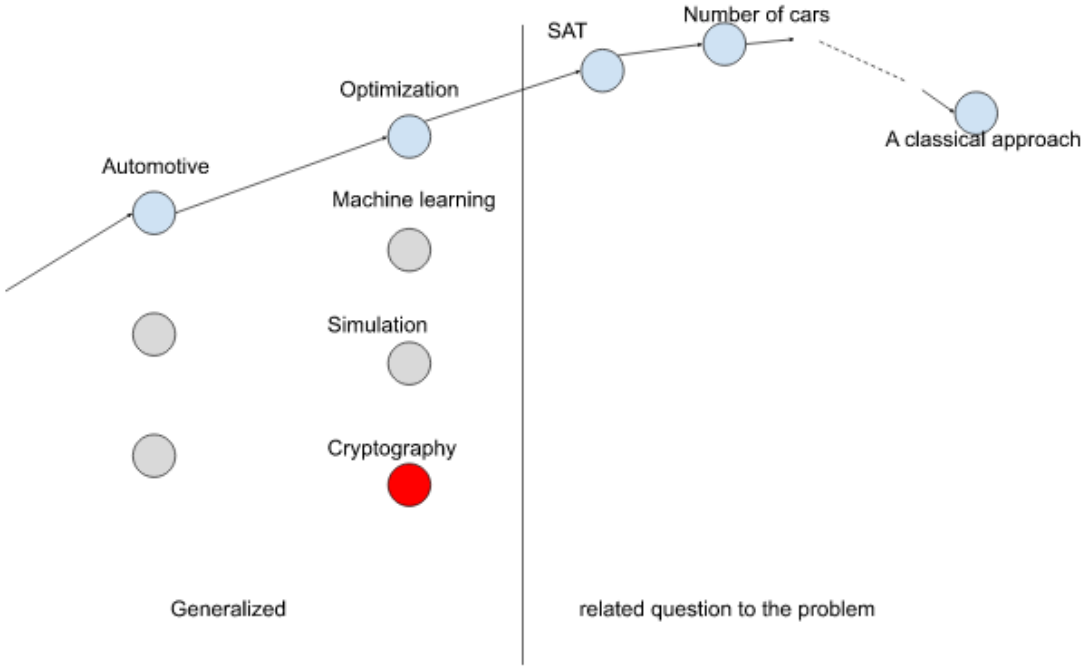


**Figure 4.3:** Tree-structure graph starting from general questions to reach a final decision

We represent the node as the current question that needs to be answered, with outgoing edges as possible answers. Figure 4.4 represents the mapping between node/edge to question/answers. The question is basically a type of node. In addition, problemClass/-Formulation/Algorithm are primarily questions but differ in that they always ask about their subject. It means that the problemClass title is always "*What/Which problem class fits your case ?*". All these nodes have a unique name and a type which defines the expected answer type, i.e., single choice or number. A question also has a description that includes the question's full title because the name is a unique small string that should not include spaces or special characters to enable creating, editing, and querying the node. The decision is a simple with the name string either quantum or classic. The other type of graph element, mainly an edge, is the answer. It includes a unique string name and a description, which is a pair of sentences to define the answer more. Later on, we added the hint field as a small paragraph describing the domain where the answer could use as part of the context comparison approach explained late in this chapter. We expect to extend the model in the future for more complex use cases, especially by defining new meta-data on problemClass/Formulation/Algorithm and creating new a type of decisions which could describe which quantum/classical method is the best to use.

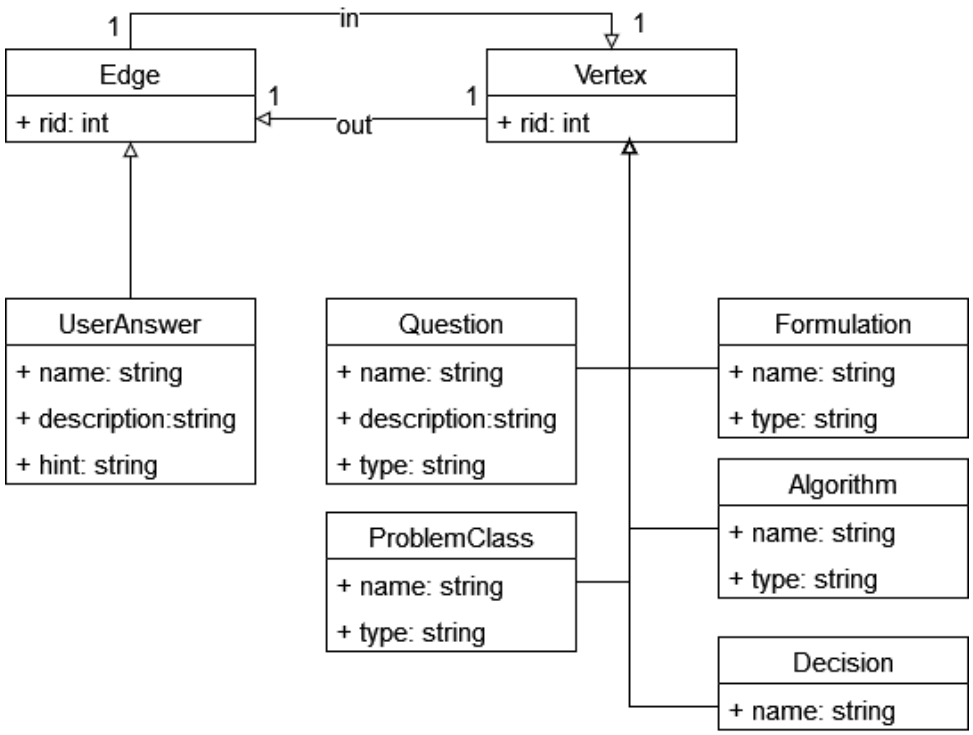


**Figure 4.4:** Class diagram of Data Schema

4.3.2.3 Mapping between Knowledge Statements and Graph Elements

In this section, we define a method to model the statements and facts related to decision-making to transform them into the graph model. The open challenge, in this case, is to translate a set of collected information into a rational model which clearly defines the relationship between them. We start by using propositional logic to define the statements, and then we demonstrate the map between statements into questions and, therefore, a graph model.

A proposition is a statement that holds a logical value of truth or false once. These propositions can be composed together with the logical connectives of AND, OR, and NOT. The statement describes something in the real world and when the statement holds truth value each time then it turns to be a fact, when this value has a justification (proof) i.e. *"The vehicle routing problem is an optimization problem"*. Thus, a recommendation is given to the users when all propositions asked the user holds truth for the user answers. Formally described $A(Q) \leftrightarrow A_1 \wedge A_2 \wedge ..... \wedge A_n | A$ is the answer of User Query $Q$ and $A_i$ a proposition that must be true. On each level -layer of knowledge - one of the possible statement must be true in order to continue to the next level in the knowledge graph, $A_{i_1} A_{i_2} A_{i_3} A_{i_4} A_{i_5}$ .i.e. The given problem should be at least in the following

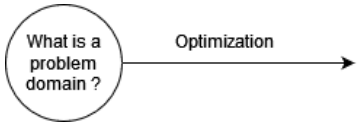


**Figure 4.5:** Simple statement modelled as a graph

domain : Optimization, Simulation, or machine learning in order to proceed to the next question.

Each statement could be turned into an open question (W-question) splitting its content into the question title and the answer. Considering "*optimization is a problem domain*" as an example, can be turned to "*What is a problem domain ?*" and the answer would be "*Optimization*". In this case, we map the question into a node with an edge going out from it to the next state, as shown in Figure 4.5 example. Inductively, this can be applied with basic logical connectives: AND, OR, NOT as shown in Figure 4.6. The AND in represents a sequence of two nodes and their answer, while the OR defines that both answers "Optimization" and "Machine learning" are outgoing answers from the same question. The NOT indicates that there is no outgoing edge labeled with the given answer.

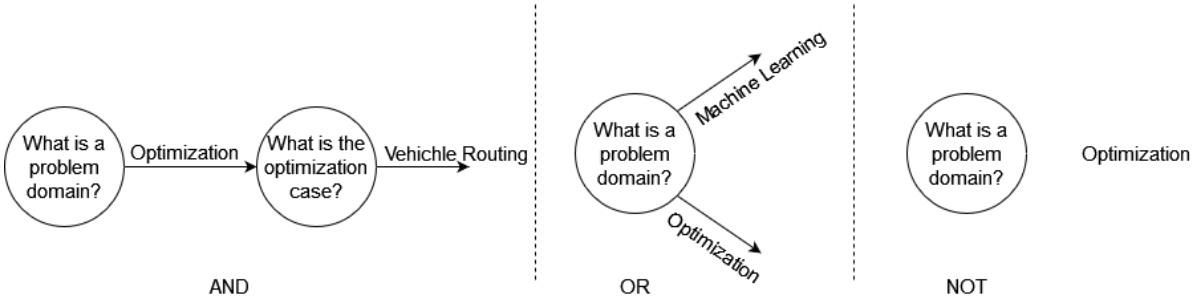


**Figure 4.6:** Mapping the logical connectives into the graph model

#### 4.3.2.4 Drawbacks of the Graph Database Model

An ideal case for getting a decision recommendation is when the end-user answers each question as a piece of information required to navigate through the graph-data base shown previously. This would be the case when asking about a popular business use case already saved in the database. However, we expect other cases the framework should handle. Following, we describe three issues that would be difficult to solve with the presented graph model.

Missing Information

The user might not have all information yet. For example, he might not be sure how many variables the problem might have. In the vehicle routing problem, the user might not be sure if he wants to consider loads in each vehicle should be delivered to each city or not. Another simple example is the absence of a use-case family. i.e. The user has an automotive industry optimization problem but does not know if it is under route optimizations or placement problems. This type of problem represents a gap in the graph as shown Figure 4.7. The dashed lines represent the usual path, but the user can not provide information to navigate sequentially, which means this information is missing. However, the user might have details not in this order and consequently needs to jump over part of the path depending on where it can provide more information. If more information is missed, it gets harder to retrieve a good recommendation. However, some steps could be optional and not necessary. As we figured out previously, each optimization problem could be modelled using SAT regardless of its context. The detection of the use case helps to detect an experiment for a similar problem done on quantum hardware and returns its results as part of the final answer. A solution to this case is to reveal each node's possible paths to reach a final decision. This can be done by navigating from a specific node through all possible paths to reach a decision node.

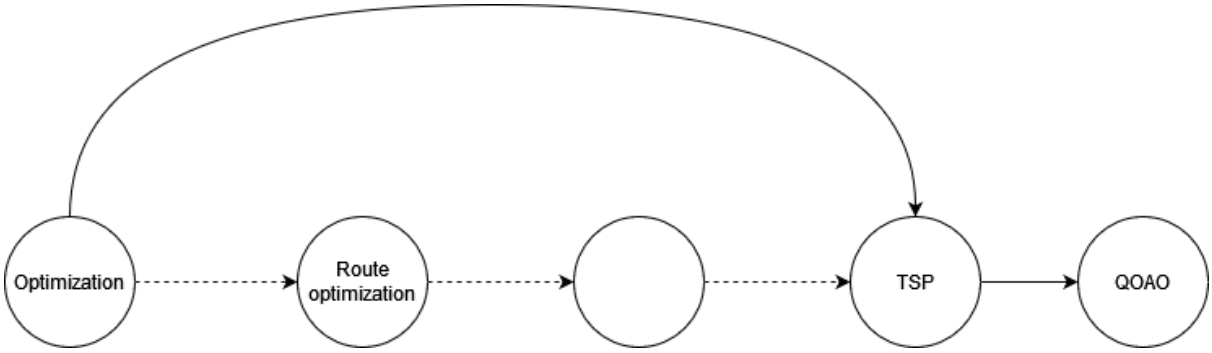


**Figure 4.7:** Missing information represent a gap between vertices in the graph

Uncertainty decision which path

Most optimization use cases could be modelled through different classes because they are NP-complete problems, as all these problems could be reduced to each other. However, In literatures [LKP21] [BBB+21], researchers selected a specific problem class related to the requirements of the use-case.i.e. vehicle route optimization problem is a graph problem, so it is easy to directly use TSP formulation since they are very similar in context, while it is still theoretically possible to use SAT. Both classes could be formulated as a QUBO

problem, which leads to a diamond shape in the graph figure of the decision. In other words, theoretically, the concert problem class used to solve the use case has no effect. However, the experiment on these models might enhance the decision-making process because it delivers practical information about performances and other benchmarks that the end-user is interested in. We provide the user two possible options : (i) to reveal all paths as in the previous case, or (ii) to provide some hints when the user can describe more about his problem. This approach is based on the context comparison explained in Section 4.4.

Use-cases still not executed on Quantum hardware

Experts recommend using quantum computing for many use cases, but there are not any experiments done yet [LKP21] [BBB+21]. The system's initial goal is to guide the user on how to solve the use case through quantum and get a final response based on practical experiences to let the user decide whether to use quantum or not. The system can not generate any recommendation to use quantum computing if there are no facts stating this case explicitly. However, we might trigger the following question: might it be possible that similar cases are helpful for users to take a decision? It can be graphically shown in Figure 4.8 as the last missing edge of the path, which leads to the final decision. One of the cases (bottom) does not have any information to lead to which formulation to use, so whether an experiment that shows if quantum computing can be used. However, the other case (Vehicle route optimization) is also modelled using TSP class which might imply that all cases modelled by the same class can get the same result. Forwarding to the next possible path seems to be a good option. It navigates back to the latest common node and tries to find the next path, ending with a decision. However, this solution suffers from many issues. First, there might be many paths from the same common point. The graph structure could not label which path is similar. Also, going back to previous knowledge in each step decreases the likelihood of getting a similar valid solution. The next path graphically does not guarantee that it should describe a similar case, especially when the knowledge base does not include many cases. To solve this problem, we used another approach in Section 4.4 to compare the contents of use cases to get the most similar known use case.

In summary, the database layer should model the knowledge to enable rational access and reasoning to derive the proper decision. We used the graph as a database model in which we transformed the propositions into questions asked sequentially to the user until they reach the latest child node, mainly quantum or classical decision. The graph model is based on the reference use case structure. However, It raised some drawbacks and issues. Therefore, we expect to extend and modify the model to adapt new complex use cases.

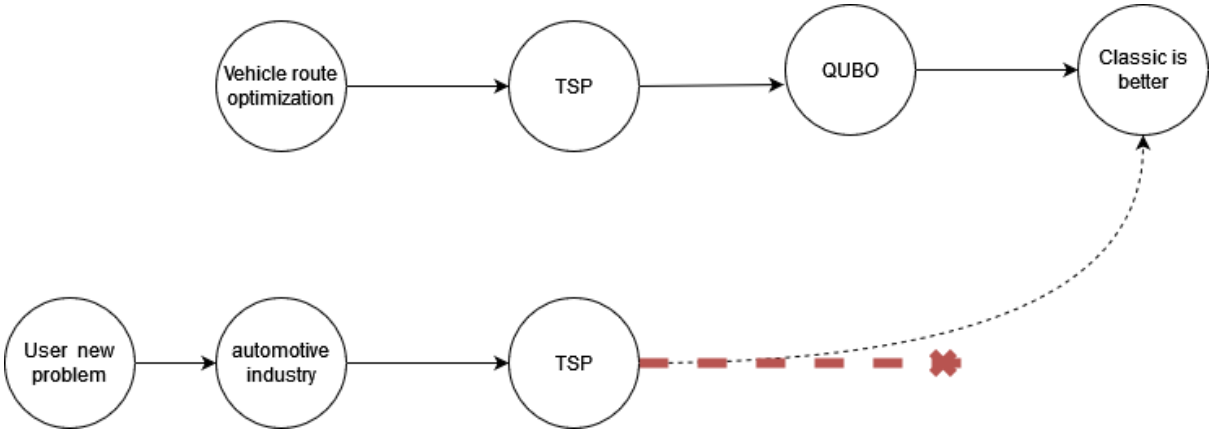


**Figure 4.8:** Reaching a dead end in one way can not hint the next similar case.

### 4.3.3 Application Layer

The application layer is responsible for bridging the user and the information from the database. Thus, it should pull information from the database and format it so the user can understand it to provide more information when needed or get a final recommendation.

We introduce QuCMixx as a microservice for handling requests from the user to retrieve information from the knowledge base and forward it to the database layer. Figure 4.9 shows the main four components of the service. The app defines the endpoints to enable easier communication and loosely coupling with other services. Based on the coming request, it calls methods from Backbone or RefCaseComparator components. Sequentially, the Backbone calls RESTpresso,–which is responsible for the communication with the database layer–, gets the answer, packs it in a defined format,–JSON format–, and delivers it back. The RefCaseComparator is in charge of receiving a description in order to compare it with the contents of available use cases or options. If the user sends a full description of a use case, it searches for a similar case. The use case's full descriptions are saved separately from the graph database. The reason is that a use case description must achieve a fixed size (number of words) to get precise results. More details are mentioned in Section 4.4. Otherwise, if the user seeks a hint, it takes the current option's hint texts on a specific question and compares them.

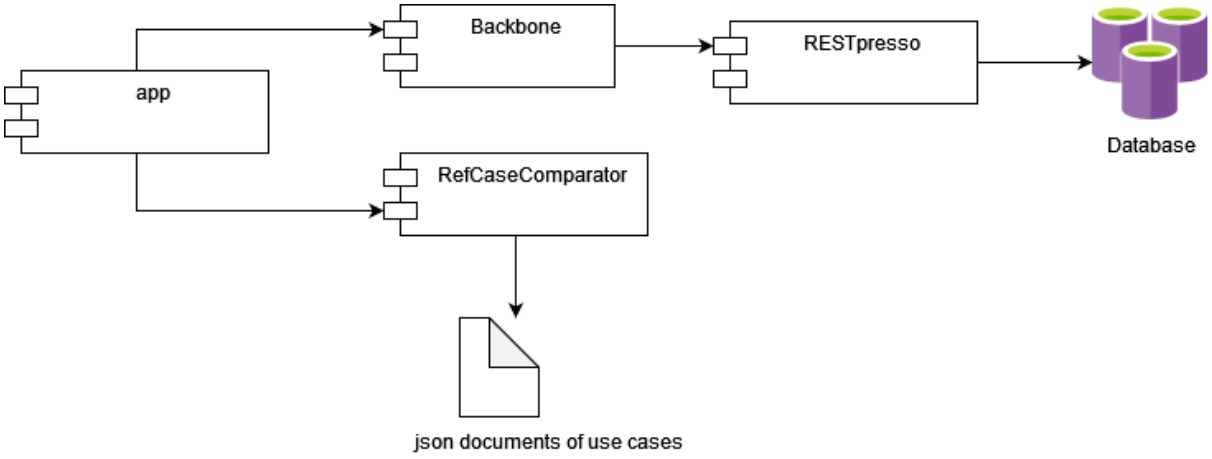


**Figure 4.9:** QuCMixx service architecture

QuCMixx needs to save the current state of the user knowledge. This means it should save the previously shown questions with their answers as well as the current returned question. Speaking on the graph database level, it should save the path starting from the root node till the last answered question. The reason behind it is to enable dynamical navigation of the graph and provide this information lately to review or decision-making process. That's why it is necessary to maintain the knowledge state during the whole session between user and the system. Session state has three main patterns [Fow12]. We review them in the following table and discuss which pattern fits the requirements of this project.

**Table 4.2:** Session States Patterns

| Client Session state | Server Session State | Database Session state |
|---|---|---|
| stores the data on the client through hidden fields, cookies, paramters | holds the data in memory between requests in serailzaible object in the server side | breaks up the data into tables and fields and stores all information in the database |

The database session state enforces modification of the structure of the database, either by creating a tabular database or a file for each user's session. In both cases, it does not fit with the dynamic modification each time the user wants to submit a new answer or go a step back. The application layer needs to control the session information directly and easily. Hence, We need to keep the data object on the server side using the server session state pattern and exchange it with the client when necessary.

In addition to QuCMixx, we create QuPie as a microservice to enable the creation, updating, and deletion of items in the database. It makes independent of the graph database if the implementation offers a user interface or not to modify the entries in the database. In addition, the endpoints can enable an automated process of adding a set of new facts. Moreover, the integration with other plugins will be easier because the endpoint enables this service's autonomy of reference, time, format, and platform. Similar to QUCMixx, the architecture of QuPie includes three main components as in Figure 4.10, starting from the app, which defines the endpoints, calling the logic function from Backbone, which starts a communication with the database through RESTacchiato.

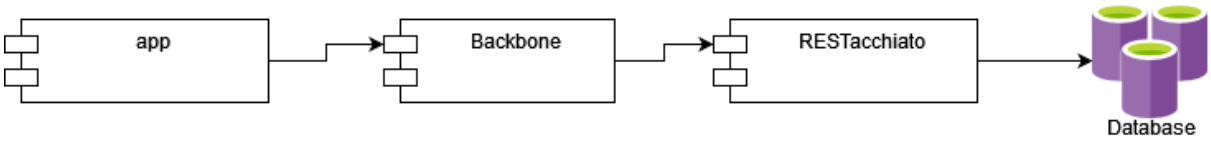


**Figure 4.10:** QuPie service Architecture

We expect that users have different tasks when using the framework. A user who is currently exploring the use cases and seeking a recommendation, which would be most of the users, maybe not interested in adding or modifying some items in the database. People who look to provide new information are researchers who have done some experiments. That's why we split the application layer into two services with separate functionalities. It helps to facilitate the scalability and improve the resiliency of the framework.

### 4.3.4 Presentation Layer

QuCface is a minimal component showing users information returned from the application layer. Also, it is responsible for retrieving the input from the user, formatting it and passing it through the defined endpoint to the application layer. Based on the returned response, it should render the UI elements each time. Therefore, it could be considered as a container which can communicate with the application layer, shows elements on each response, and could fit in every UI implementation.

### 4.3.5 Communication & Integration

The communication between the different services has an impact factor of the sustainability and reliability of the system. That's why the communication style should be

stable, which can be achieved by minimizing the dependencies between the services so that they are loosely connected. The framework implicates the REST architecture [WPR10], which defines the REST-API art of communication between components. Each service from each layer defines a set of fixed endpoints, the data needed to proceed, and the output form. In this case, each service can easily be integrated directly with other frameworks. Furthermore, the client layer could be adapted with any UI plugin by using the available tools to render elements. The application layer could be packaged to import the functions defined in the Backbone components of both services directly without communicating through the endpoints. This could be helpful when integrating with QuAntiL project mentioned in the non-functional requirements. The implementation art enhances the concept of easy integration, which we discuss in Chapter 5.

The overall architecture of the framework is shown in Figure 4.11. The user can access the framework through QuCface service, communicating with QuCMixx to pass collected answers. QuCMixx has access to a set of documented use cases for comparison requests or can navigate with simple answers through the graph database in the deepest layer of the framework. QuPie is a separate service which performs CUD (Create, Update, Delete) operations on the database. The endpoints of this service tend to be integrated with an entire pipeline of quantum computing applications that might include searching, decision, and executing through a set of services.

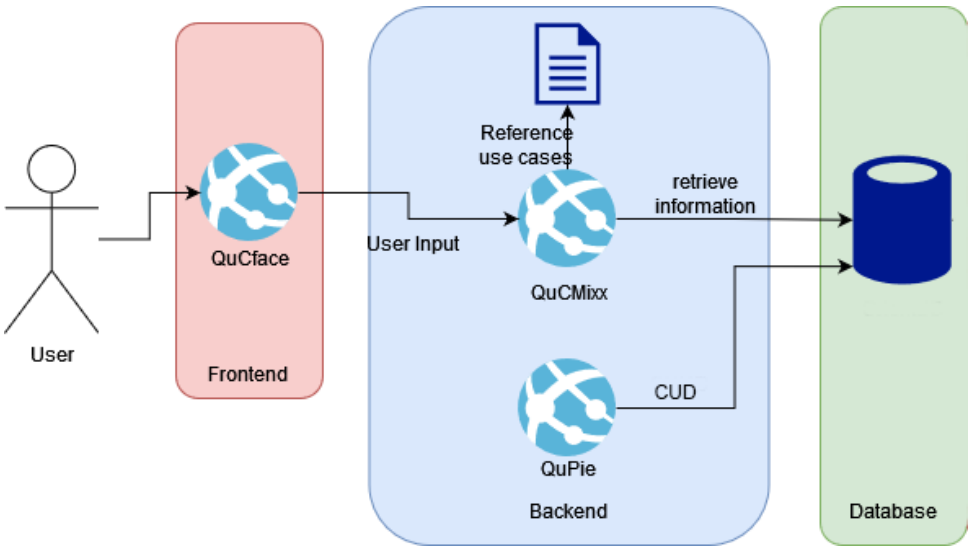


**Figure 4.11:** Overall architecture of QuCSplit framework

## 4.4 Use cases comparison using NLP

The graph knowledge approach has raised some issues that are hard to solve because it defines a deterministic path between the states. This occurs when the desired path deviates from the defined ones in the database. The experiments [BBB+21] [LKP21] showed that the results could not prove that quantum computers always overcome the classical methods. Each use case might produce different and unexpected results, so we cannot predict or derive generic rules when the quantum is better to use. However, researchers [BBB+21] [LKP21] were able to group the use cases based on the potential implementation on quantum computers. Thus, it is likely that cases with the same implementation or mathematical formulation to produce similar results thanks to the separation of each of the four levels defined in the reference use case pattern. One example in [LKP21] is that the Vehicle Routing Problem and the Robot Production Planning can be modelled using TSP. If experiments on VRP showed that quantum computing overcomes the classical methods' results, then it would be likely to have the same decision in the other case because the step from case context to the problem class eliminates all related details to an abstracted model that is valid for different cases. Another example is that authors [GKS22] model the Vehicle Configuration for general purposes in any configuration problem, not restricted to vehicles only. This means that any case description that includes a configuration optimization can use this model and is likely to get similar results. To conclude, in case of the absence of a deterministic decision, similar cases might support the user as well to get a recommendation about the given case based on the matching with other cases. The matching is based on the agreement of contents summarized from publications of the use cases in four different layers: Problem domain, problem class, formulation, and algorithm.

Similar to the recommendation system based on movies' plot [BP14], this approach consists of four main steps, to provide a comparing solution between use cases description, that analyze the texts and detect the similarity ratio:

**Keywords extraction:** It extracts a set of keywords based on statistical features in the text. The current unsupervised methods do not guarantee the extraction of special terms related to the problem class, formulation, or the intended algorithm to use, such as TSP,QAOA... etc. That's why we explicitly search for these terms by defining a list for each level of the reference case pattern, filled with terms that could also be retrieved from benchmarks defined in [FRH+22].

**Weight computation:** The resulting vector of each description represents the one-hot encoding of the keywords extracted. The match between two cases in deeper levels showed that it increases the probability of getting similar results. That's why we defined a scale from 1(less important) to 4 (more important) to represent the importance of word occurrences based on their topic. Figure 4.12 shows the

defined measure. The occurrence of words in the domain of the problem has the lowest grade, while Keywords detected in the algorithm level has the best grade because it implicates the use of the same class and formulation since experiments were able to define a mapping between those levels. Consequently, the match between descriptions in thev formulation subject includes using the same problem class but could try another algorithm or implementation. Hence, the scale follows the order of the four levels in the reference case pattern, which is also the same in the graph model.

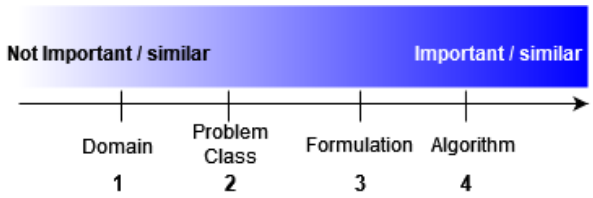


**Figure 4.12:** The importance scale of words appear in any description

**Matrix of cases $\times$ keywords:** To allow the comparison between all descriptions, we order all vectors in one matrix with indexing all keywords found. In this case, each vector has the encoding of any keyword occurrence retrieved from all documents.

**Simialrity computation:** We use the cosine similarity between the vectors and the given description from the user. Since the vectors do not include negative coordinates, which means they do not describe the opposite of each other, the cosine similarity measure returns a value between 0 (not related at all) and 1 (identical). Thus, we consider the biggest value as the most similar use case to the requested one. The increased weights affects the convergence between vectors, which enhance the similarity between use cases. Figure 4.13 is an example of this case. It shows the impact of weight of the second dimension of the vectors. $Z = (1, 2), u = (1, 1)$ are normalized consequently to $(\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}})$ and $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. The vector $Z$ which has a greater weight is closer to vector $v$ than $u$.

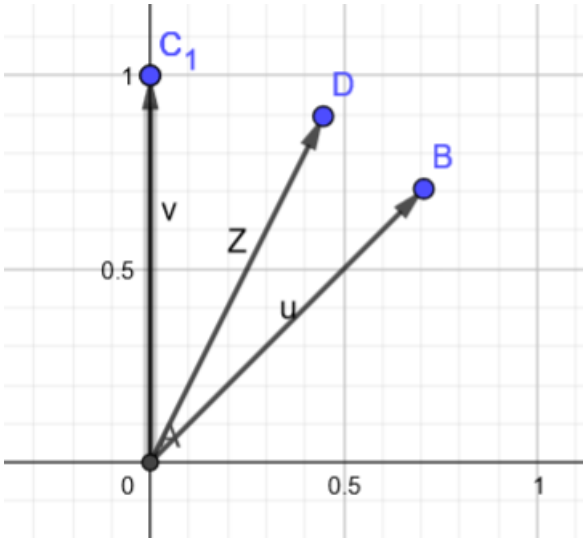


**Figure 4.13:** The weights impact on vectors convergence

We integrate this approach in the application layer in two functionalities: (i) If the user wants to submit a full description of the use case he is searching for once and do want to get with multiple interactions with the framework. This could also be helpful by linking with some text extraction tool that summarizes the description of a specific use case and wants to retrieve some recommendations. (ii) If the user is unsure which option to answer a specific question and wants to get a hint, which one is potentially helpful for the case. The hint feature compares the small descriptions saved in the database for each answer; thus, we adapt the method in each case to deal with either bigger text or some small descriptions.

To conclude, The comparison based on the contents of the use cases' descriptions aims to support the graph database model and bridge the gap with non-found or new cases. The navigation in the graph depends on each time the user answer, and reaching a final decision can not retrieve which use case was asked in the first place (stateless database). Thus, we cannot generate similar results if we try to gather all questions with answers to each possible path from the root to the decision. Moreover, The graph model saves the facts minimised as questions and the comparator would stick to the words used in the database entries. Therefore, the approach requires rich text, including more details, to check similar cases. That's why each document should include a description of the four levels : (i) Use case context, (ii) Problem Class, (iii) formulation, and (iv) algorithm. If the description does not contain all these points, the user can still get some results, but it would be less accurate.

# 5 Implementation

The implementation realizes the presented approach in the previous chapter. Thanks to the microservice architecture pattern, we can independently use various programming languages and platforms for each service. We briefly explain the frameworks, libraries, and platforms we used to implement this work. A prototype of this project is published on GitHub[1].

## 5.1 OrientDB

OrientDB [20a] is the first Multi-Model Open Source NoSQL DBMS that the creation of graph and document databases in one high-performance operational database. The release is associated always with a server that can run locally or easily be deployed in the cloud. In addition, it is associated with an environment studio to control the databases through a friendly user interface. This feature was very helpful since it can visualize the graph database. In Figure 5.1, we see the graph editor of the studio. The user can dynamically interact with the database by dragging the nodes, changing the display information, or changing the distance or color configuration.

Although OrientDB is designed to handle NoSQL databases, it saves and modifies the entries in a relational database using SQL. Each graph database is initialized with two main classes: vertex(V) and edge(E). Each class has a table in the relational model in which the instances are recorded. The table includes standard columns required for the metamodel of the graph, like in and out fields in the Edge table, which refer to instances from Vertices to describe the source and destination of an edge. In addition, users can inherit new classes from each type to define a customized model that includes further fields (properties) and rules.

To enable integration with the application layer, OrientDB offers three ways for communication: (i) Native binary remote that calls methods directly against the TCP/IP socket using the binary protocol, (ii) REST APIs that support communication based on

[1] https://github.com/mohamad-altaweel/QUCSplit

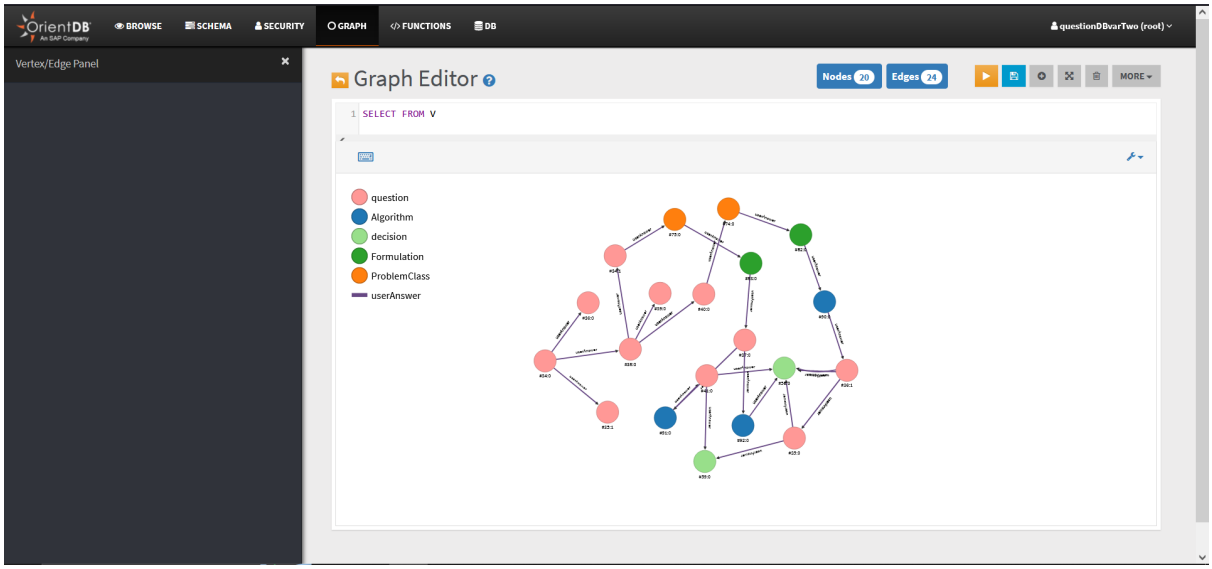


**Figure 5.1:** OrientDB studio visualizes the graph database with key labels of the model

REST architecture using the HTTP protocol, and (iii) Java wrapped, as a layer that binds the native Java driver [20a]. In addition, many plugins integrate OrientDB in the code directly, such as PyOrient for python and PHP Orient for PHP. However, using those drivers as part of the code in the application layer will enforce a tight coupling with the database. That's why we restrict the defined architecture by using the REST APIs. The server lets the user define endpoints for REST API communication. Each endpoint triggers a function defined in SQL or JavaScript to perform a specific transaction on the database, also with passed parameters through the API. This enables a more straightforward implementation of the communication part from the application layer, which needs only to send the requests and format the parameters as defined.

OrientDB includes contributions from different people and communities [2]. However, the releases suffer from different bugs and issues, which are still open. The release 3.2.8 we used has problems with executing JavaScript code from an endpoint. That's why we switch to SQL functions and had to implement the validation of parameters and logic as part of the application component because the integrated JavaScript would have offered more flexibility in dealing with the graph database directly as an object-oriented model, not as a relational table.

[2] https://github.com/orientechnologies/orientdb

## 5.2 Flask

Flask [22d] is a micro web framework written in Python. It does not require any tool or library and has no database layer or any other pre-existing components. However, it is compatible to integrate with new components or extensions as part of the flask application. For this reason, it is more fit to implement simple microservices than other frameworks like Django, laravel, or Springboot, which already define a complex architecture for advanced uses of web applications.

The flask is imported into the app component in QuCMixx and QuPie services, which defines the REST endpoint and runs the application as a flask web service. Otherwise, other components can be used as python packages and imported into other scripts. In this case, we provide both low-level code integration and API communication. QuCMixx returns a JSON response on each request, including two main keys: header, a unique word that indicates the type of response and expected content, and context, which includes all details of the response. This allows the external component to easily predict the response based on the value of the header key. We list the main endpoints of QuCMixx and the header values of each response with their description.

**Table 5.1:** Endpoints of QuCMixx

| Endpoint | returned header | Meaning |
|---|---|---|
| /<question>/<answer> | question | gets a question |
| /<question>/<answer> | decision | gets a final decision |
| /<question>/<answer> | NotFound | passed answer was not found |
| /<question>/<answer> | traversed | returns all paths from current point to decision |
| /back | – | redirects to the previous question endpoint |
| /OneTextAnswer | Similar | return the similar case based on given text |
| /OneTextAnswer | NotEnoughWords | needs more words than given |
| /hint/<question> | hint | returns a hint of possible options |
| /hint/<question> | NotEnoughWords | needs more words than given |

The easier access offered by OrientDB minimized the role of QuPie as a service to perform modification of the database and keep it as a tool to do these operations only when no communication directly with the database is desired.

## 5.3 YAKE

YAKE! is an unsupervised automated keyword extraction approach that relies on text statistical data gathered from individual documents to choose the text's most crucial keywords. It does not depend on dictionaries, external corpora, text size, language, or domain, nor require training on a specific collection of documents. This method is compared with ten state-of-the-art unsupervised techniques (TF.IDF, KP-Miner, RAKE, TextRank, SingleRank, ExpandRank, TopicRank, TopicalPageRank, PositionRank, and MultipartiteRank) and one supervised approach in order to show its strengths and significance (KEA). Experimental findings demonstrate that the approach greatly outperforms state-of-the-art methods under various collections of varying sizes, languages, or topics. These results were obtained on top of twenty datasets [CMP+20] [CMP+18b] [CMP+18a]. We use YAKE in the first step of the use cases' descriptions comparison approach by getting a list of keywords from the text. The user can define the number of required keywords to extract. The output is a list of words; each is associated with a score denoting the relevance of the word. The lower the score, the more relevant the keyword is. Depending on the text size we expect and experimenting with the results obtained from YAKE, it has shown that one needs twenty keywords to be extracted for a full description of a use case and ten keywords from a hint text of a possible answer. However, more similar cases could complicate the process and require more keywords to differentiate between them. Unfortunately, The framework still faces a lack of resources to test the comparison on more complex cases, which leaves an open issue to review and discuss.

## 5.4 Bootstrap

Bootstrap [22a] is an open-source CSS framework which develops responsive user interfaces. It consists of HTML, CSS and JavaScript design templates for many UI components such as forms, navigation, button...etc. This project aims to develop a minimal functional interface component that can demonstrate the features delivered from the application layer. That's why we used Bootstrap without needing further and advanced frameworks or libraries.

## 5.5 Docker

Docker [22c] is an open platform for packaging, deploying, and running applications. It enables running a software application in an isolated environment as a lightweight virtual

machine called the container. This environment can be managed and customized through the platform. In this case, one can run more containers simultaneously. Furthermore, docker provides to create a virtual network which lets these containers communicate with each other as if they are connected to the WAN. Docker intends to facilitate the deployment and release and reduce the gap between development and operation.

The microservice architecture of QuCSplit provides easy use of docker. Each defined service can be packaged in a docker container, especially with the different dependencies that each part demands. OrientDB can run on a host that includes Java Runtime Environment (JRE), and each application service needs python installed with the required packages such as flask, YAKE, and ordered_set. However, to run any application services, we must ensure that the orientdb service is available to establish a successful connection. QUCface interface component does include only HTML code which does not require any particular demands and can be run easily from any browser. For this purpose, we use the predefined virtual network from docker to connect all these containers and retrieve their virtual IP addresses.

# 6 Evaluation

The evaluation of this project ensures that it has reached the defined goal in the problem statement. We follow constructive research, which aims to create tools, methods and techniques that can practically solve a problem and present a contribution to the main topic. Therefore, the evaluation of the approach could be done by testing if the introduced tool can achieve the defined requirements. The V-Model [LL10] can organize the test activities as part of the software development process, as shown in Figure 6.1. It starts by formalizing the problem that states how we can construct a method to decide to split between quantum or classical uses of different applications. Then, it derives the requirements for a software system to make such a decision and characterizes the system architecture. After finishing the implementation, the testing process is started. Each level aims to test the opposite requirements, which helps to build the evaluation hierarchically. Thus, Integration tests aim to ensure reliable communication between parts of the system, which helps to deliver the functional features at the acceptance test level and consequently solves the original problem. We ignored mentioning the Unit test level explicitly since it is integrated with the implementation and implicated with higher levels of tests. In this chapter, we briefly explain each level to demonstrate this project's achievement and discuss the introduced approach's open issues instead of explaining the technical level and implementation of testing but only focus on the proof of the concept using this framework.

## 6.1 Integration test

The database and application layer provides its main functionalities through fixed endpoints. Application layer services are also dependent on APIs offered by the database. To enable a systematic review and test of all endpoints, we list the endpoints provided from the database in Table 6.1 and the application layer endpoints, listed in Table 5.1, with their dependencies in Table 6.2. In this way, we can define the test paths that can cover all integration points to reach full coverage of the integration test.

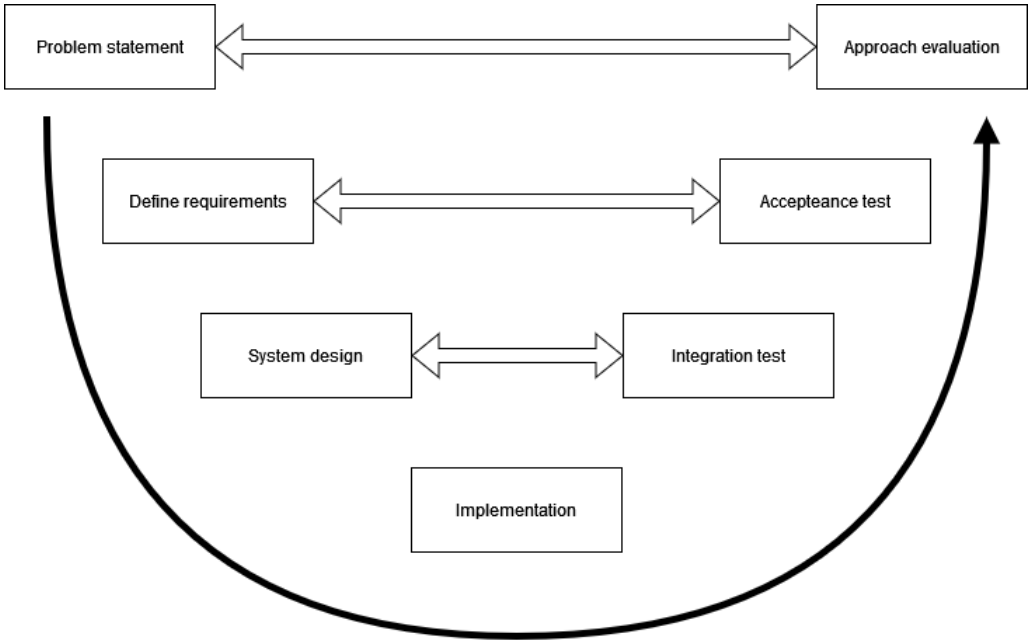


**Figure 6.1:** V-Model of QuCSplit approach

**Table 6.1:** Endpoints provided by database layer

| Endpoint | Required Parameters | Functionality |
|---|---|---|
| QuestionExists | question name | Return the question entry details |
| GetPossibleAnswers | question name | Return all outgoing edges from the question |
| AnswerExists | Answer name (Edge label) | Return the answer (edge) details |
| GetNextNode | Answer ID | Get the destination node of this edge |
| getEdge | destination & source questions | Get the edge connecting those nodes |
| getDecisions | – | Retrieve all nodes from class decision |
| getVertexFromID | Question ID | Return the vertex entry details |
| navigate | Questions'name | Get all paths from this question to decision node |
| sql | SQL command | Execute the given SQL command |

| Endpoint | depends on |
|---|---|
| <question>/<answer> | QuestionExists, GetPossibleAnswers, GetNextNode, getDecisions, navigate, getVertexFromID, getEdge, AnswerExists |
| back | <question>/<answer> |
| OneTextAnswer | – |
| hint/<question> | GetPossibleAnswers & |
| QuPie endpoints | sql |

**Table 6.2:** Endpoints provided by database layer

## 6.2 Acceptance test

The acceptance test per ISTQB [20b] checks whether the system can achieve the requirements and satisfy the acceptance criteria so can be accepted by the stakeholders. Therefore, we list each requirement in Section 4.1 and which features have been implemented for it in order to define the test cases for the acceptance test.

**R1. Knowledge database construction:** The OrientDB database, represented as the database layer, should enable the creation, modification and deletion of any entry that represents information that helps to reach a decision. Hence, the database should accept any piece of information that is transformed into a propositional statement as a question-answer model. QuPie service supports handling necessary operations on the database for this purpose.

**R2. Decision recommendation service:** QuCMixx is part of the application layer that should receive user requests as answers of a question. This interactive process aims to retrieve the necessary information to provide a decision to use quantum or classical computers. Another way to provide information can be done by submitting a full-text description. The user should get the most similar case to the given one with the accuracy ratio.

**R3. User inteface component:** QuCface should show the questions and final decisions received from the application layer. It is responsible to return a useful feedback to the user in all cases. Overall, when all parts of the system can deliver the expected output correctly, then the following test cases should be satisfied:

- Each time the user should get the first question in the database firstly.
- If the user answers a question, then it should get either a new question or a final decision.

- If user needs hint, then it should get a hint to a possible answer when he sends a small text requesting for hint
- If user can not proceed, noIdea option should retrieve all paths from the current node to at least one of both decision.
- If the user sends a full description of the desired use case, the system should return the most similar case including its description, link of the publication, accuracy ratio.

## 6.3 Discussion

In conclusion, the research question investigates a method to support vendors and researchers by splitting the use of quantum and classical computing into different use cases. This project shows that using a decision tree based on prior experiments' statements could support the selection process when they are structured in a graph database. In addition, content comparison using NLP methods has shown a promising result to get similar use cases as a good recommendation for the user to review. Finally, we discuss the significant drawbacks of the project's approach.

### 6.1 Approach's Drawbacks

The database assumes the boolean type of all answers. Thus, the user must select a choice from a fixed list of options. Even questions which expect to get a number are turned into a logical comparison which proceeds to the next step if the given number is smaller or greater than the boundary points. That means that the current model can not deal with questions that need to execute some calculations. A possible future work would be to investigate cases in which decisions are not based only on a sequence of choices but even let the user answer open questions, perform some calculations, and may be linked to more than one use case. The hard challenge here is that any change in the data requires much change in the database schema. Another issue is that we can not add new use cases only when an experiment on the quantum computer has been done and shows the performance of it because it describes the final decision on which technology to use. That means all potential cases mentioned in [LKP21] which seek potential use of quantum computers can not be added unless executed. This downgrades the extension of the knowledge base and delays the deployment of this framework.

Keywords extraction and text matching should be evaluated with a dataset of many instances to review its performance in solving a current problem and assess it as an

efficient solution. Due to the lack of experiments and published reference cases, we could not be able to conduct an assessment on our approach. The tests were done in the current use cases, which is considered not enough yet. This forms a critical point to this approach because the project can not state if this method can successfully produce accurate results, but only depending on an evaluation study case. For this reason, we consider this method as an initial approach, which must be further evaluated and improved when more cases are available. NLP provides more methods for topic selection and texts comparison, also using machine learning and neural networks.

## 6.1 Technical's Drawbacks

The communication between the application layer and the database is done by searching for nodes based on their Name field. Since it deals in the end with entries written in SQL, Name should include no space or special character and must be unique for this Node. This makes it difficult to create and modify nodes without prior knowledge of all current entries in the database to avoid duplication and misuse of the database. In addition, the application layer is not stateless, which makes it hard to provide scalability to the system. The endpoint can interact with the user when he submits a question not starting from the root node, but then the session from previous application instances would be lost.

# 7 Conclusion and Future Work

The last chapter gives a brief summary of this project. In addition, it presents the open points for future projects in this subject to improve this approach in upcoming research.

## 7.1 Summary

In this project, we introduce a new method which helps to decide whether to use quantum or classical computing in different use cases. It is based on a decision tree which is constructed on information gathered from previous experiments. The knowledge base of decisions is implemented as a graph database accessible through a microservice that can interact with the user primarily through a set of questions and answers. In addition, we used NLP methods for keywords extraction to compare the contents of the use cases to provide decisions for similar cases. The practical part of the project is implemented through a microservice-based system that could be easily extended with further functionalities, as well as the integration with other tools and frameworks intended to provide easy and reliable use of quantum technology.

## 7.2 Future Works

- The project investigated researches that attempt to solve the problems entirely using quantum computing. One complex challenge is to review hybrid approaches and introduce a new type of decisions that might demand extra information.
- The database schema saves only simple textual information about the entries. A further improvement is to extend the schema to get new questions that require a kind of calculations or are linked with more than one use case.
- The similarity approach must be evaluated with a set of use cases. The accurate results of this approach could enable the full replacement of the decision rule because it does not need any transformation from textual to graphical representation.

# List of Figures

# List of Abbreivation

# Bibliography


[20a] *Orientdb Documentation*. version 3.0. 2020 (cit. on pp. 16, 49, 50).

[20b] *Standard Glossary of Terms used in Software Testing*. version 3.2. 2020 (cit. on p. 57).

[22a] *Bootstrap documentation*. version 5.2. 2022 (cit. on p. 52).

[22b] *D-Wave Documentation*. 2022 (cit. on p. 29).

[22c] *Docker documentation*. version 4.12.0. 2022 (cit. on p. 52).

[22d] *Flask documentation*. version 2.2.x. 2022 (cit. on p. 51).

[22e] *Quantum Application Lifecycle Management (QuAntiL)*. https://quantil.readthedocs.io/en/latest/. 2022 (cit. on pp. 9, 28).

[22f] *Solving Problems with Quantum Samplers*. https://docs.dwavesys.com/docs/latest/c_gs_3.html. 2022 (cit. on p. 31).

[22g] *What is a Knowledge Graph?* https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/". [Online; accessed 5-September-2022]. 2022 (cit. on p. 16).

[ABA+22] U. Azad, B. K. Behera, E. A. Ahmed, P. K. Panigrahi, A. Farouk. "Solving Vehicle Routing Problem Using Quantum Approximate Optimization Algorithm." In: *IEEE Transactions on Intelligent Transportation Systems* (2022), pp. 1–10. DOI: 10.1109/TITS.2022.3172241 (cit. on p. 30).

[ASMD19] E. Amador-Domínguez, E. Serrano, D. Manrique, J. F. De Paz. "Prediction and Decision-Making in Intelligent Environments Supported by Knowledge Graphs, A Systematic Review." In: *Sensors* 19.8 (2019). ISSN: 1424-8220. DOI: 10.3390/s19081774. URL: https://www.mdpi.com/1424-8220/19/8/1774 (cit. on p. 25).

[BBB+21] A. Bayerstadler, G. Becquin, J. Binder, T. Botter, H. Ehm, T. Ehmer, M. Erdmann, N. Gaus, P. Harbach, M. Hess, et al. "Industry quantum computing applications." In: *EPJ Quantum Technology* 8.1 (2021), p. 25 (cit. on pp. 7, 21, 22, 28, 35, 39, 40, 45).

[BHO+22] M. Biondi, A. Heid, I. Ostojic, N. Henke, L. Pautasso, N. Mohr, L. Wester, R. Zemmel. *Quantum computing use cases are getting real—what you need to know*. https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/quantum-computing-use-cases-are-getting-real-what-you-need-to-know". [Online; accessed 3-September-2022]. 2022 (cit. on pp. 7, 13, 15).

[BMMW21] T. Bennett, E. Matwiejew, S. Marsh, J. B. Wang. "Quantum walk-based vehicle routing optimisation." In: *Frontiers in Physics* 9 (2021), p. 730856 (cit. on pp. 30–32).

[BP14] S. Bergamaschi, L. Po. "Comparing LDA and LSA topic models for content-based movie recommendation systems." In: *International conference on web information systems and technologies*. Springer. 2014, pp. 247–263 (cit. on pp. 25, 26, 45).

[Bra97] G. Brassard. "Searching a Quantum Phone Book." In: *Science* 275.5300 (1997), pp. 627–628. DOI: 10.1126/science.275.5300.627. eprint: https://www.science.org/doi/pdf/10.1126/science.275.5300.627. URL: https://www.science.org/doi/abs/10.1126/science.275.5300.627 (cit. on p. 13).

[CBS+19] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, J. M. Gambetta. "Validating quantum computers using randomized model circuits." In: *Phys. Rev. A* 100 (3 Sept. 2019), p. 032328. DOI: 10.1103/PhysRevA.100.032328. URL: https://link.aps.org/doi/10.1103/PhysRevA.100.032328 (cit. on p. 23).

[CMP+18a] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, A. Jatowt. "A text feature based automatic keyword extraction method for single documents." In: *European conference on information retrieval*. Springer. 2018, pp. 684–691 (cit. on p. 52).

[CMP+18b] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, A. Jatowt. "Yake! collection-independent automatic keyword extractor." In: *European Conference on Information Retrieval*. Springer. 2018, pp. 806–810 (cit. on p. 52).

[CMP+20] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, A. Jatowt. "YAKE! Keyword extraction from single documents using multiple local features." In: *Information Sciences* 509 (2020), pp. 257–289 (cit. on p. 52).

[FGL+21] D. Fitzek, T. Ghandriz, L. Laine, M. Granath, A. F. Kockum. *Applying quantum approximate optimization to the heterogeneous vehicle routing problem*. 2021. DOI: 10.48550/ARXIV.2110.06799. URL: https://arxiv.org/abs/2110.06799 (cit. on p. 30).

[FL05] J. Forrest, R. Lougee-Heimer. “CBC user guide.” In: *Emerging theory, methods, and applications*. INFORMS, 2005, pp. 257–277 (cit. on p. 29).

[FNAD20] N. Firoozeh, A. Nazarenko, F. Alizon, B. Daille. “Keyword extraction: Issues and methods.” In: *Natural Language Engineering* 26.3 (2020), pp. 259–291. DOI: 10.1017/S1351324919000457 (cit. on p. 26).

[Fow12] M. Fowler. *Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch*. Addison-Wesley, 2012 (cit. on pp. 32, 42).

[FRH+22] J. R. Finžgar, P. Ross, L. Hölscher, J. Klepsch, A. Luckow. *QUARK: A Framework for Quantum Computing Application Benchmarking*. 2022. DOI: 10.48550/ARXIV.2202.03028. URL: https://arxiv.org/abs/2202.03028 (cit. on pp. 23, 24, 29, 45).

[GKS22] A. Glos, A. Kundu, Ö. Salehi. *Optimizing the Production of Test Vehicles using Hybrid Constrained Quantum Annealing*. 2022. DOI: 10.48550/ARXIV.2203.15421. URL: https://arxiv.org/abs/2203.15421 (cit. on pp. 29, 45).

[GS18] D. J. Griffiths, D. F. Schroeter. *Introduction to quantum mechanics*. Cambridge university press, 2018 (cit. on p. 11).

[Gur21] Gurobi Optimization LLC. *Gurobi Optimizer Reference Manual*. 2021 (cit. on p. 29).

[Hid21] J. D. Hidary. “Dirac Notation.” In: *Quantum Computing: An Applied Approach*. Springer, 2021, pp. 377–381 (cit. on p. 12).

[IWT+19] H. Irie, G. Wongpaisarnsin, M. Terabe, A. Miki, S. Taguchi. “Quantum annealing of vehicle routing problem with time, state and capacity.” In: *International Workshop on Quantum Technology and Optimization Problems*. Springer. 2019, pp. 145–156 (cit. on pp. 30, 31).

[JL03] R. Jozsa, N. Linden. “On the Role of Entanglement in Quantum-Computational Speed-Up.” In: *Proceedings: Mathematical, Physical and Engineering Sciences* 459.2036 (2003), pp. 2011–2032. ISSN: 13645021. URL: http://www.jstor.org/stable/3560059 (visited on 08/28/2022) (cit. on p. 12).

[KJS18] B. Kamiński, M. Jakubczyk, P. Szufel. “A framework for sensitivity analysis of decision trees.” In: *Central European journal of operations research* 26.1 (2018), pp. 135–159 (cit. on p. 17).

[LBF+20] F. Leymann, J. Barzen, M. Falkenthal, D. Vietz, B. Weder, K. Wild. “Quantum in the Cloud: Application Potentials and Research Opportunities.” In: *Proceedings of the 10$^{th}$ International Conference on Cloud Computing and Services Science (CLOSER 2020)*. SciTePress, May 2020, pp. 9–24 (cit. on p. 13).

[LKP21] A. Luckow, J. Klepsch, J. Pichlmeier. “Quantum computing: Towards industry reference problems.” In: *Digitale Welt* 5.2 (2021), pp. 38–45 (cit. on pp. 21–23, 35, 39, 40, 45, 58).

[LL10] J. Ludewig, H. Lichter. *Software Engineering – Grundlagen, Menschen, Prozesse, Techniken; 2., überarb., aktualisierte und erg. Aufl.* Heidelberg: dpunkt.verlag, 2010, XXI, 665 S.. : Ill., graph. Darst. ISBN: 978-3-89864-662-8. URL: https://publications.rwth-aachen.de/record/96850 (cit. on p. 55).

[Man08] C. D. Manning. *Introduction to information retrieval*. Syngress Publishing, 2008 (cit. on p. 18).

[Ond20] L. P. Ondrej Burkacky Niko Mohr. *Will quantum computing drive the automotive future?* https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/will-quantum-computing-drive-the-automotive-future". [Online; accessed 5-September-2022]. 2020 (cit. on p. 15).

[Qui87] J. Quinlan. “Simplifying decision trees.” In: *International Journal of Man-Machine Studies* 27.3 (1987), pp. 221–234. ISSN: 0020-7373. DOI: https://doi.org/10.1016/S0020-7373(87)80053-6. URL: https://www.sciencedirect.com/science/article/pii/S0020737387800536 (cit. on p. 17).

[RV97] P. Resnick, H. R. Varian. “Recommender systems.” In: *Communications of the ACM* 40.3 (1997), pp. 56–58 (cit. on p. 25).

[WPJ+21] A. Wack, H. Paik, A. Javadi-Abhari, P. Jurcevic, I. Faro, J. M. Gambetta, B. R. Johnson. *Quality, Speed, and Scale: three key attributes to measure the performance of near-term quantum computers*. 2021. DOI: 10.48550/ARXIV.2110.14108. URL: https://arxiv.org/abs/2110.14108 (cit. on p. 23).

[WPR10] J. Webber, S. Parastatidis, I. Robinson. *REST in practice: Hypermedia and systems architecture*. " O’Reilly Media, Inc.", 2010 (cit. on p. 44).

All links were last followed on October 20, 2022.

**Declaration**

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature