Patterns for Quantum Error Handling

Martin Beisel, Johanna Barzen, Frank Leymann,
Felix Truger, Benjamin Weder, Vladimir Yussupov

Institute of Architecture of Application Systems,
University of Stuttgart, Germany,
[firstname.lastname]@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{Beisel2022_ErrorHandlingPatterns,
  author    = {Beisel, Martin and Barzen, Johanna and Leymann, Frank and
               Truger, Felix and Weder, Benjamin and Yussupov, Vladimir},
  title     = {{Patterns for Quantum Error Handling}},
  booktitle = {Proceedings of the 14\textsuperscript{th} International
               Conference on Pervasive Patterns and Applications (PATTERNS
               2022)},
  year      = {2022},
  month     = apr,
  pages     = {22-30},
  publisher = {Xpert Publishing Services (XPS)},
  Isbn      = {978-1-61208-953-9}
}
```

Patterns for Quantum Error Handling

Martin Beisel, Johanna Barzen, Frank Leymann, Felix Truger, Benjamin Weder, Vladimir Yussupov
Institute of Architecture of Application Systems, University of Stuttgart
Universitätsstrasse 38, 70569 Stuttgart, Germany
{firstname.lastname}@iaas.uni-stuttgart.de

***Abstract*—The capabilities of current quantum computers are limited by their high error rates. Thus, reducing the impact of these errors is one of the crucial challenges for the successful execution of quantum algorithms. For this purpose, various error handling methods have been proposed, ranging from error correction codes that detect and correct errors during the execution on a quantum device to post-processing techniques that mitigate errors classically. As these methods have different requirements and advantages, developers need to have a thorough understanding of them, to be able to select a suitable error handling method for their scenario. In this work, we present three new patterns for quantum error handling, describing proven solution strategies in a well-structured manner and integrate them into an existing quantum computing pattern language.**

***Keywords*-Quantum Computing; Pattern Language; Error Handling; Error Mitigation; Error Correction.**

I. Introduction

Recent advances in the development of quantum devices resulted in the emergence of publicly available quantum computers [1]. Quantum computers are expected to solve certain problems, e.g., in chemistry [2] or combinatorial optimization [3], more efficiently than any classical computer. This is possible because quantum computers can exploit quantum mechanical effects, such as superposition and entanglement, to gain a computational advantage. However, the capabilities of the current generation of quantum computers are limited by a variety of factors, e.g., the low number of qubits and high error rates [1][4]. Due to the high number of errors caused by different error sources, e.g., error-prone gate and measurement operations, the execution result's accuracy is limited [5]. To deal with these errors, different error handling techniques have been proposed. Error correction codes, such as Shor's 9-qubit code [6], can be used to detect and correct occurring errors [7][8][9]. However, error correction requires a significant number of additional quantum resources. Thus, so-called error mitigation methods have been developed that require little to no additional quantum resources. These methods focus on the reduction of the negative impact caused by certain error types, e.g., Tensor Product Noise Model (TPNM) [10] and Fixed Identity Insertion Method (FIIM) [11] for measurement errors and gate errors, respectively. To incorporate any of these error handling methods into quantum applications, quantum software engineers need to understand their concepts, so they can select suitable ones for the case at hand.

A well-established approach for the description and structuring of proven solutions for reoccurring problems was presented by Alexander et al. [12] in the form of *patterns*. Each pattern describes a problem and its context and forces. Then, a proven solution for the problem is presented in an abstract manner, making the pattern applicable to different scenarios. Multiple patterns of the same domain can be combined in a *pattern language*. A pattern language for quantum computing has been introduced by Leymann [13]. Although it has been continuously expanded since its introduction [14][15][16], the quantum computing pattern language does not contain any patterns for the handling of quantum errors yet.

In this work, we extend the quantum computing pattern language by introducing three new patterns that describe well-established solutions for the handling of quantum errors. By documenting these proven solutions in an easy-to-understand and well-structured manner, we provide knowledge about quantum error handling to a broader audience. The ERROR CORRECTION pattern describes how to detect and correct quantum errors during the quantum computation. As this approach is not always feasible, two more patterns are introduced that focus on the mitigation of the impact of occurring errors. First, the READOUT ERROR MITIGATION pattern describes, how the impact of measurement errors can be reduced. Second, the GATE ERROR MITIGATION pattern presents proven solutions for the mitigation of gate errors that occur during the quantum circuit execution.

This paper is structured as follows: Section II introduces fundamental terms to establish a common vocabulary and describes the used pattern format. Then, Section III presents the error handling patterns in detail. In Section V, the related work is discussed, and Section VI concludes this work.

II. Fundamentals and Pattern Structure

In this section, we present our pattern format and establish the guidelines for the pattern authoring process. Further, we provide the fundamental terms related to quantum error handling that establish a common vocabulary.

A. Pattern Format & Authoring Method

The pattern format is derived from previous work on quantum computing patterns [14][15][16] and other best practices used by researchers [12][13][17][18][19][20]. A pattern is identified by its *name* and a mnemonic *icon*. First, the *problem* solved by the pattern is briefly described in form of a short question. Then a detailed description of the pattern's *context* and its *forces* is presented. The *solution* section describes a possible solution with a corresponding sketch. The *result* paragraph explains the context following the application of the solution and discusses possible consequences. Afterwards, one or multiple *examples* of the previously introduced solution are explained textually and visually. In the *related patterns* section, the relationship of the pattern to other patterns within the pattern language is described. Finally, the *known uses* section lists implementations of the pattern.

For the identification of the quantum error handling patterns, we analyzed state-of-the-art approaches in scientific literature. Recurring solution strategies were collected, analyzed, and ultimately compiled into the quantum error handling patterns. Due to the lack of currently available code fragments implementing the error handling patterns, identifying a rich collection of concrete solutions remains future work. These can then be integrated into a future quantum computing solution language, facilitating the patterns' application [21].

B. Fundamental Terms

Quantum Device: A quantum device is a gate-based quantum computer, e.g., IBM's, Google's, or IonQ's quantum computers. Quantum devices can execute *quantum circuits* that implement *quantum algorithms*. Typically, device access is provided via the cloud. Some providers offer free access to small devices [22], however, access to state-of-the-art devices is costly and can scale with the number of executed quantum circuits and their size [23]. Further, the limited number of available devices, in combination with the high demand for usage, can lead to queue times. Due to the fragility of coherent quantum states, quantum devices are error-prone. This is particularly true for currently available Noisy Intermediate-Scale Quantum (NISQ) devices.

Quantum Algorithm: A quantum algorithm is an algorithm that can be executed on a quantum device and typically makes use of quantum mechanical effects to achieve a computational advantage over its classical counterpart. The gate-based representation of a quantum algorithm is a quantum circuit. Most currently used quantum algorithms are hybrid quantum-classical algorithms that consist of a classical and a quantum part. Typical examples are Variational Quantum Algorithms (VQAs), such as Variational Quantum Eigensolver (VQE) [24] and the Quantum Approximate Optimization Algorithm (QAOA) [25]. VQAs employ shallow parameterized quantum circuits that are optimized classically, to perform meaningful computations on current NISQ devices [26].

Quantum Circuit: A quantum circuit is a model for quantum computation. Each step of the computation is modeled by a *quantum gate*. At the end of each circuit a *measurement* is performed, to retrieve the result of the quantum computation. Due to the probabilistic nature of quantum computing, the circuit has to be executed multiple times to obtain a reliable probability distribution. The depth of a circuit is the number of layers of 1- or 2-qubit gates in which parallel operations are performed on disjoint qubits. Its width is defined as the number of qubits involved in the computation.

Quantum State: A quantum state describes the current state of one or multiple qubits. While classical bits can be in the states 0 and 1, qubits can be in corresponding $|0\rangle$ and $|1\rangle$ states, however, they can also be in a superposition, i.e., a combination of both states. A quantum state can be measured to obtain a probability distribution for each possible state.

Ancilla Qubit: Ancilla qubits follow the concept of classical ancilla bits. They are additional qubits that are typically used temporarily to store information or achieve a specific goal, e.g., they can store entangled states.

Quantum Gate: Quantum gates are the elementary operations of a quantum circuit that can be executed on a quantum device's qubits. Quantum gates are unitary operations, which can be described mathematically by unitary matrices. Quantum devices only support a limited gate set, which is usually restricted to a small number of 1- or 2-qubit gates [27]. Furthermore, quantum devices can execute quantum gates only with limited accuracy, resulting in so-called gate errors [5]. As these errors can occur with every execution of a gate, they continuously accumulate during the quantum circuit execution.

Measurement: *Readout* or *observation* are common synonyms for measurement. The quantum state prepared by a quantum circuit is sampled by performing a measurement operation. By measuring a qubit, its state is collapsed and can not be restored. Therefore, a measurement operation is not a quantum gate. Moreover, measurement times are significant in comparison to gate times, causing delays that amplify the devices's decoherence [5]. Hence, measurements are one of the main error sources of a quantum device [28].

Quantum Error: Quantum errors, also known as *Noise*, are one of the key limitations in the current era of quantum computing. The capabilities of quantum devices are limited by the error-prone and highly fragile quantum states that are used for computation, as the occurrence of too many errors makes the measurement result unusable. First, the aforementioned quantum gate errors can occur with every execution of a gate. Thereby, the error rates of multi-qubit gates are particularly high [27]. These errors account for a significant part of the overall error and limit a quantum circuit's depth. Second, unintended bit-flips occurring during the measurement, lead to incorrect measurement results. To execute a quantum circuit on a quantum device, the circuit needs to be compiled for the device's gate set and qubit connectivity mapping [29]. As each of the qubits usually only has a direct connection to a small subset of the other qubits, a lot of SWAP operations may be required to establish multi-qubit operations foreseen in the uncompiled quantum circuit [4]. Such swap operations lead to additional 2-qubit gates and increase the circuit depth. Thus, designing a circuit for a specific device or vice versa can prevent errors. Additionally, qubits can unintentionally influence the state of other qubits [5]. This so-called crosstalk is difficult to predict and can further increase the overall error. Moreover, quantum states can decohere, meaning that after a device-dependent amount of time, they decay irreversibly due to environmental influences. Hence, all quantum gates and measurement operations have to be performed before the state decoheres, directly limiting a circuit's maximum depth.

III. Patterns for Quantum Error Handling

In this section, we introduce three new patterns focusing on error handling for quantum devices, namely ERROR CORRECTION, GATE ERROR MITIGATION, and READOUT ERROR MITIGATION. These patterns focus on the prevention and reduction of errors occurring during the quantum circuit execution and extend the existing quantum computing pattern language [13][14][16]. First, we present the error handling pattern category in the context of the quantum computing patterns and then introduce each new pattern in detail.

A. *Quantum Computing Patterns for Error Handling*

The quantum computing patterns form a pattern language supporting quantum software engineers in developing quantum applications. As most quantum applications are hybrid [4][30] the focus is on hybrid-classical algorithms. The quantum computing patterns capture reoccurring problems in the domain of quantum computing and provide proven solutions for them. Figure 1 shows the basic structure of a hybrid algorithm [4] and an overview of the quantum computing pattern language. The process starts off, by pre-processing data on a classical computer. A typical example is the preparation of the data required for state preparation. The state preparation routine prepares the quantum computer's initial state, e.g., a uniform superposition can be created, or the previously prepared data can be encoded into the initial state. This can be done by employing one of the quantum state or data encoding patterns. The prepared quantum state can then be manipulated by applying unitary transformations. These perform the quantum algorithm's operations. Typical operations are explained in detail in the unitary transformation patterns [13]. Best practices for realizing quantum algorithms are described in the program flow patterns [16]. The last step performed on the quantum computer is the measurement operation. Thereby, the quantum algorithm's final state is retrieved in the form of a probability distribution. The measurement result can then be post-processed, e.g., performing continued fraction expansion for Shor's algorithm [31]. In the case of a VQA, the hybrid algorithm has a loop. Thereby, the result is incorporated into the next iteration unless its termination condition is fulfilled. Thus, the quantum computing patterns cover the entire cycle of a hybrid quantum algorithm.

However, in their current state, the quantum computing patterns do not address the handling of errors. Since quantum errors are one of the major factors limiting quantum computing, particularly in the current NISQ era, reducing their negative impact is essential. Due to the variety of error sources and hardware limitations, different solution strategies have evolved. The ERROR CORRECTION pattern focuses on the in-flight repair of computational errors. It enables fault-tolerant large-scale quantum computing by detecting and fixing errors immediately during the circuit execution. However, error correction requires a large number of quantum resources that are not available on current quantum devices yet. A NISQ-compatible alternative to error correction is error mitigation. Error mitigation tolerates the occurrence of errors and focuses on the reduction of their impact. In particular, the READOUT ERROR MITIGATION and GATE ERROR MITIGATION pattern document two different kinds of error mitigation, focusing on the reduction of the impact of measurement and gate errors.

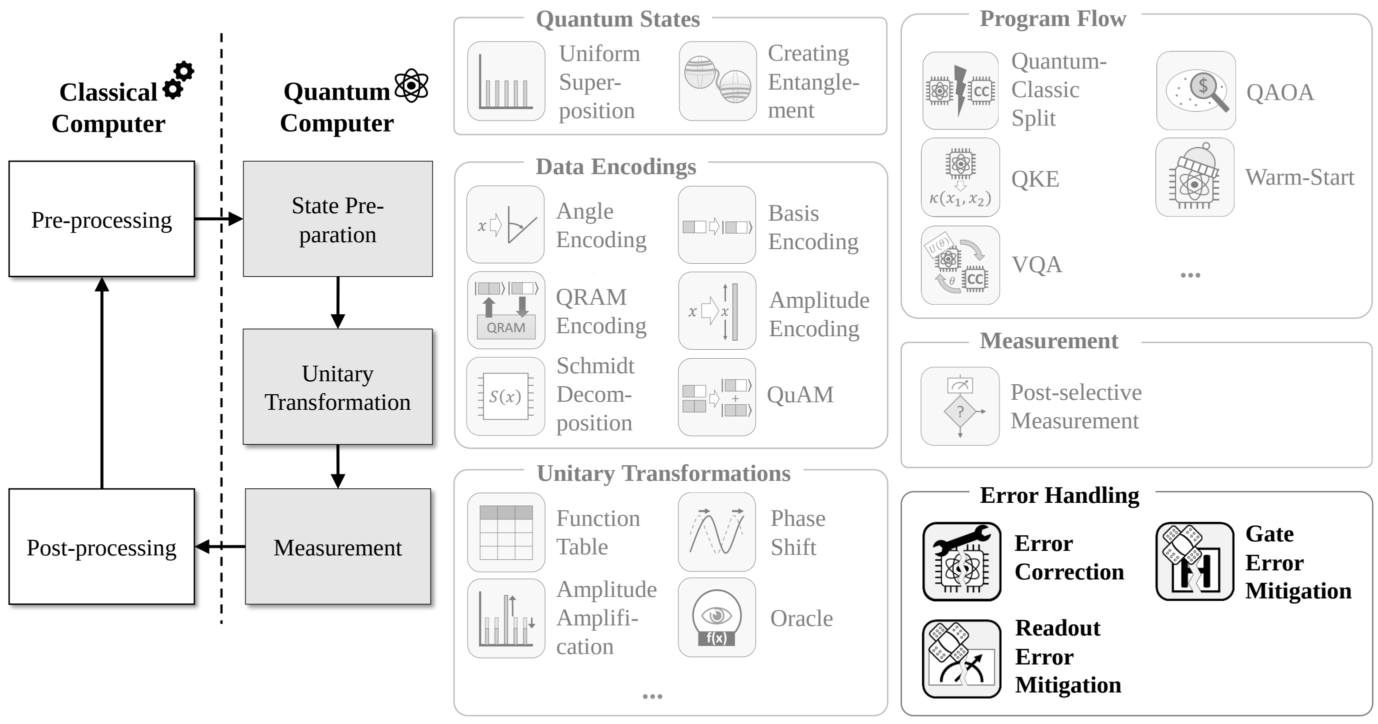

Figure 1. Overview of a hybrid algorithm's typical structure and existing and new (in bold) quantum computing patterns, extends [14][16].

B. Error Correction Pattern

How to detect and correct errors occurring during the execution of a quantum circuit?

Context: A quantum algorithm needs to be run on a quantum device. The quantum device's performance is limited by various error sources, such as gate errors and crosstalk. The prevention of these errors enables the execution of large-scale quantum algorithms for real-world problems.

Forces: Quantum devices unavoidably cause a certain amount of errors due to the fragility of coherent quantum states [7][32]. Furthermore, contrary to classical bits, qubits can not be copied [33]. Hence, classical error correction can not be used for quantum computers and new quantum-specific methods need to be developed. However, these methods can be costly in terms of quantum resources, as they require a large number of additional qubits and quantum gates.

To enable scalable quantum computing for real-world problems, all kinds of errors occurring in quantum devices need to be detected and corrected. In general, the correction of errors is preferred over their mitigation, since even minor remaining post-mitigation errors slowly stack up during the computation and ultimately lead to an imprecise result.

Solution: Detect and correct quantum errors using quantum error correction codes [7][34][35][36], which are added to the executed circuit. With these correction codes, many physical qubits are combined into one logical qubit. As a result of this bundling, errors in the original qubit can be first detected and then corrected [4]. Figure 2a depicts a solution sketch showcasing the general building blocks of a quantum error correction procedure. The shown instance applies an error correction code that can detect and fix bit-flip errors in the computational basis. For the correction of errors from other sources, similar processes can be applied. First, the *ancilla coupling* is created, by encoding the state $|\psi\rangle$ of a single physical qubit into multiple ancilla qubits. These qubits now hold the logical qubit's data and are called *data qubits* in the following. Next, some unitary transformation is applied to the logical qubit, possibly resulting in an error. In order to *detect* an error, additional ancilla qubits are employed to check the parity of the data qubits. Based on the discovered syndrome, the error-free state can be recovered in the *recovery phase*. Note that the process is assumed to only have errors at the unitary transformation step, which is denoted by the error indicator. Further, the number and type of detectable errors depends on the applied error correction code.

Result: When applying quantum error correction, computational errors can be prevented, enabling error-free systems of logical qubits. Thus, error correction is making fault-tolerant quantum computation feasible. The good scalability of error correction, enables the accurate execution of large algorithms.

Examples: Figure 2b illustrates the application of a 3-qubit variant of the aforementioned error code for multiple qubits. Each of the physical qubits P1 to P4 is transformed into a logical qubit consisting of five physical qubits. Three of these five physical qubits are being used as data qubits and two of them are being used for the detection and recovery process. Further, the 1- and 2-qubit gates G1 to G4 need to be realized by the subroutines S1 to S4, which prepare the data qubits. The resulting errors can then be corrected by individually applying error correction routines for each of the logical qubits.

Related Patterns: Instead of preventing quantum errors, the READOUT ERROR MITIGATION and GATE ERROR MITIGATION patterns focus on reducing the errors' negative impact on results. This pattern can be applied to quantum algorithms, e.g., QAOA pattern.

Known Uses: Laflamme et al. [8] show a 5-qubit error correction code that can protect a qubit against general 1-qubit errors. Shor's 9-qubit code can protect a qubit against single bit-flip and phase-flip errors [6]. Further, a variety of different quantum error correction codes have been presented in the literature [7][34][35][36].

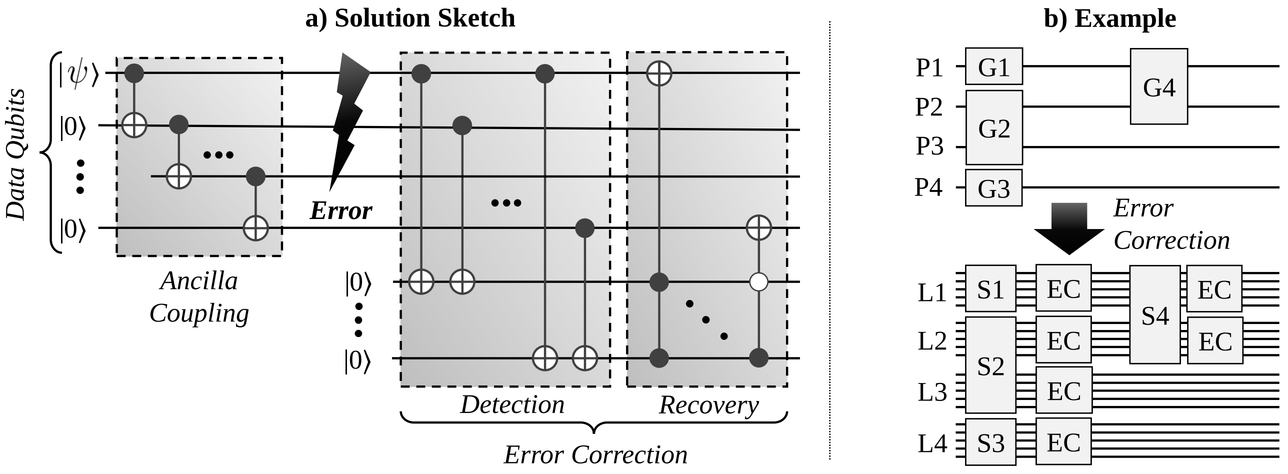

Figure 2. Solution sketch for the ERROR CORRECTION pattern and example of error correction with a 3-qubit bit-flip code for a 4-qubit circuit [4][7].

C. Readout Error Mitigation Pattern

How to reduce the impact of erroneous measurements such that the measured result is closer to the intended quantum state?

Context: A NISQ-compatible quantum algorithm, e.g., QAOA or VQE, needs to be run on a quantum device. The device's decoherence times are short and the measurement operations are error-prone. Hence, the measured probability distribution is inaccurate, even when the measured quantum state is accurate. Thus, the negative impact of readout errors needs to be mitigated to obtain a precise measurement result.

Forces: The measurement times of quantum computers in the NISQ era are significant in comparison to their decoherence times [37]. Therefore, the measurements are highly error-prone and often are among the main error sources [10]. Due to the limited capabilities of current NISQ devices, a minimal number of additional qubits and quantum gates shall be used for the mitigation of readout errors. Further, a quantum device's measurement error rates change over time, thus the Readout Error Mitigation (REM) needs to be adaptive.

Solution: Mitigate the impact of readout errors by applying a REM method. The mitigation method is performed after the circuit execution and adjusts the measured probability distribution. The resulting mitigated probability distribution is a more accurate representation of the intended quantum state. A solution sketch for the application of REM is shown in Figure 3a. First, the quantum circuit is implemented and executed. Then the resulting probability distribution is improved based on measurement characteristics collected for the quantum device. These characteristics are typically obtained by separately running so-called calibration circuits. Alternatively, adapted instances of the implemented circuit can be run to obtain additional information about the measurement properties.

Result: REM can reduce the impact of errors caused by measurement operations. The resulting, more precise probability distributions make NISQ devices more suitable for real-world use cases. However, additional classical processing is necessary, which can significantly increase the runtime and classical resource requirements, as not all mitigation methods scale well with the number of qubits. Generally, data provenance can be employed to increase the efficiency of frequently occurring REM tasks, e.g., when executing a VQA.

Examples: Figure 3b illustrates the steps of the Static Invert-and-Measure (SIM) [28] technique. First, multiple slightly adapted instances of the circuit are created. Thereby, bit-flips are added right before the circuit's measurement operations. This helps to detect erroneous measurements because readout error rates are typically higher when measuring a qubit in the $|1\rangle$ state than when measuring it in the $|0\rangle$ state [28]. Once all circuits are executed, the measurement results are processed, returning the mitigated probability distribution. Figure 3c shows the typical process of a calibration matrix-based mitigation method. Multiple shallow calibration circuits are generated and executed. The resulting probability distributions give information about the device's readout error rates. These error rates are then incorporated into a so-called calibration matrix, which can be used to mitigate readout errors. For example, this can be done by multiplying the inverse of the calibration matrix with the circuit's measurement result.

Related Patterns: The GATE ERROR MITIGATION pattern can be used in combination with this pattern for more extensive mitigation. This pattern can be applied to hybrid quantum algorithms, e.g., VQE or QAOA pattern. This pattern commonly uses BASIS ENCODING pattern to generate calibration circuits.

Known Uses: Various REM methods, e.g., calibration matrix-based [10][37][38][39] or bit-flip-based [28][40][41], have been introduced in the literature. Moreover, recent work introduces a deep learning-based REM method [42].

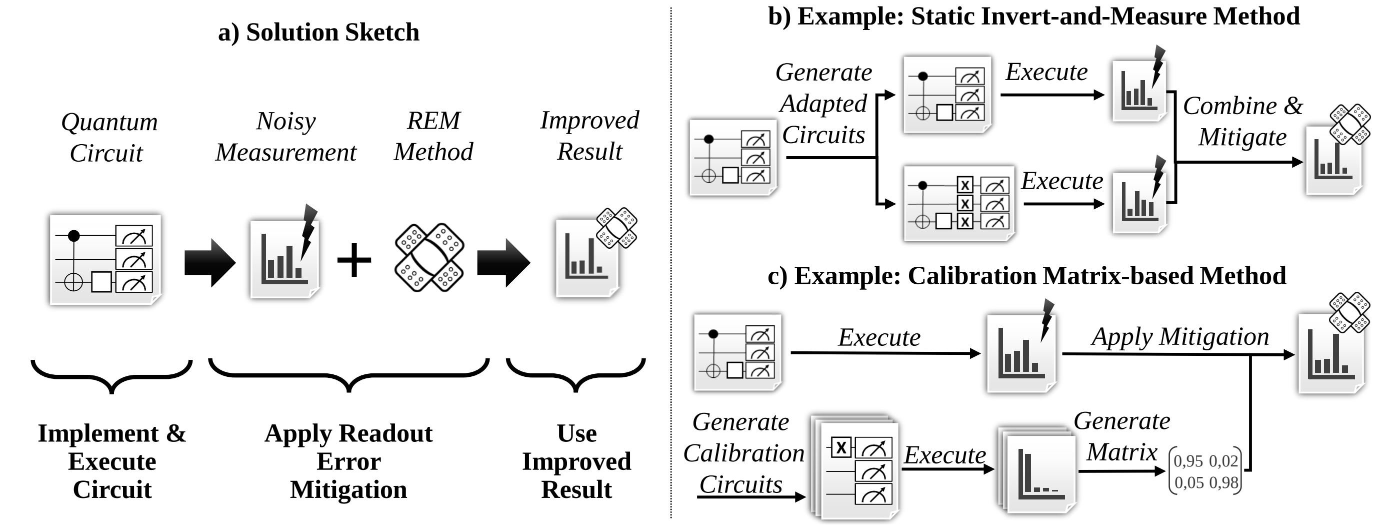
a) Solution Sketch
Quantum Circuit
Noisy Measurement
REM Method
Improved Result
Implement & Execute Circuit
Apply Readout Error Mitigation
Use Improved Result
b) Example: Static Invert-and-Measure Method
Generate Adapted Circuits
Execute
Execute
Combine & Mitigate
c) Example: Calibration Matrix-based Method
Execute
Apply Mitigation
Generate Calibration Circuits
Execute
Generate Matrix
0,95 0,02
0,05 0,98

Figure 3. Solution sketch for the READOUT ERROR MITIGATION pattern and two example processes for bit-flip- and calibration matrix-based methods.

D. Gate Error Mitigation Pattern

How to reduce the negative impact of noisy gate executions such that the pre-measurement state is closer to the expected error-free state?

Context: A NISQ-compatible quantum algorithm, e.g., VQE, needs to be run on a quantum device. The device's gate implementations are error-prone, causing errors in the quantum computation. To obtain precise results for the executed algorithm, the measured state needs to be computed accurately. Thus, it is crucial to mitigate the effects of gate errors.

Forces: The execution of gates on current NISQ devices is not perfectly accurate. Hence, every execution of a gate causes a minor error. These errors keep accumulating, eventually making large computations impossible. The pulses used for the implementation of gate operations can be controlled on many quantum devices [43][44]. Therefore, custom pulse schedules can be used to individually calibrate gates. Furthermore, the capabilities of current quantum devices are limited, e.g., the number of qubits and the decoherence times are bound. Thus, minimal additional quantum resources, such as gates and qubits, shall be used for error mitigation.

Solution: Mitigate the impact of gate errors by applying a Gate Error Mitigation (GEM) method. The mitigation of gate errors has to be performed before the execution of the quantum circuit, as occurring errors otherwise accumulate during the computation, making it difficult to retrace them. The resulting pre-measurement quantum state is closer to the expected error-free state, therefore, providing more accurate measurement results. Figure 4a depicts a solution sketch for GEM. First, the circuit is implemented. Afterwards, a GEM method is applied, modifying the circuit, to generate a more precise implementation for the selected device. The circuit modifications can range from simple gate additions over custom gate pulse adjustments to full circuit rewrites based on Machine Learning (ML). Next, the improved circuit is executed on the quantum device. Finally, the improved measurement result can be evaluated to obtain a more precise solution.

Result: GEM can significantly reduce the impact of errors caused by erroneous gate executions. As a consequence, the state computed by the quantum algorithm is closer to the expected error-free quantum state and a more precise algorithm result can be obtained. However, the mitigation process may induce additional quantum gates into the circuit or require classical pre-processing to calculate optimal device calibrations, e.g., gate pulse calibrations. Generally, GEM methods can be used in combination with other error mitigation methods, such as REM to reduce the overall error further.

Examples: Figure 4b shows the process of a typical gate addition-based method. These methods mitigate gate errors by adding additional gates to the quantum circuit that balance out gate errors. The initial quantum circuit is modified by adding specific gates for each error-prone operation. Hence, the depth of the circuit increases significantly. Therefore, the device's decoherence times need to be kept in mind, as otherwise, the mitigation might decrease the result quality.

Figure 4c depicts the typical process of a pulse calibration method. First, the pulse calibrations for the device are generated, e.g., it is determined which frequency is perfect to perform a bit-flip operation on a specific qubit. Once all required frequencies are determined, the information can be incorporated into the quantum circuit. When executing the modified circuit, the custom pulse calibrations will now be used instead of the default values. More precise pulse calibrations make gate executions more accurate, thus, decreasing gate error rates and increasing the solution's precision.

Related Patterns: This pattern can be used in combination with the READOUT ERROR MITIGATION pattern to further reduce the overall error. This pattern can be applied to hybrid quantum algorithms, e.g., VQE or QAOA pattern.

Known Uses: Several circuit adjustment methods, e.g., FIIM or Random Identity Insertion (RIIM), are presented in the literature [11][45][46]. Further, machine learning-based circuit adjustment methods have been introduced, e.g., Noise-Aware Circuit Learning (NACL) [47]. Moreover, pulse modification-based GEM methods have been presented [48][49].

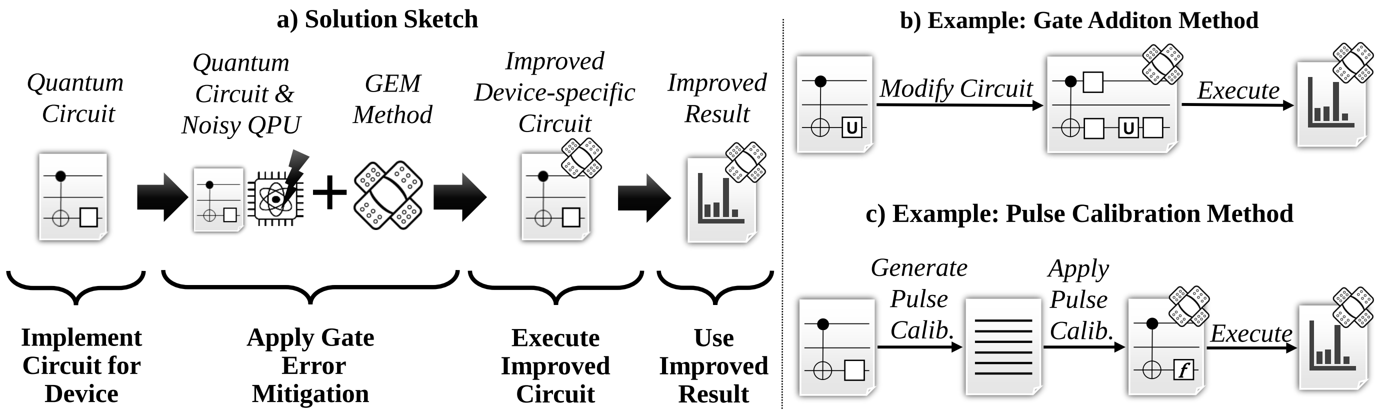
a) Solution Sketch
Quantum Circuit
Quantum Circuit & Noisy QPU
GEM Method
Improved Device-specific Circuit
Improved Result
Implement Circuit for Device
Apply Gate Error Mitigation
Execute Improved Circuit
Use Improved Result
b) Example: Gate Additon Method
Modify Circuit
Execute
U
c) Example: Pulse Calibration Method
Generate Pulse Calib.
Apply Pulse Calib.
Execute
f

Figure 4. Solution sketch for the GATE ERROR MITIGATION pattern and two example processes for a gate addition and a pulse calibration method.

IV. Pattern Validation and Discussion

In software engineering, a pattern's validity can be confirmed by showing the existence of a large enough number of real-world occurrences [17][19]. For each presented quantum error handling pattern there exist multiple distinct real-world occurrences as shown in Table I.

The Error Correction pattern is implemented on a basic level by Laflamme's 5-qubit code [8] and Shor's 9-qubit code [6] which protect a single qubit against general 1-qubit errors, and bit- and phase-flips, respectively. Consequently, more advanced methods emerged that rely on such basic error correction implementations, e.g., subsystem or topological error correction codes [7]. Some examples include Bacon-Shor (BS) [50], Calderbank-Shor-Steane (CSS) [35][51][52] and surface codes [36].

The Readout Error Mitigation pattern's most used implementations rely on calibration matrices. Prominent examples are TPNM [10], Continuous Time Markov Processes (CTMP) [10], Matrix-free Measurement Mitigation (M3) [39], and Diagonal Detector Overlapping Tomography (DDOT) [53]. In contrast, the SIM [28], Adaptive Invert-and-Measure (AIM) [28], and Bit-Flip Averaging (BFA) [40] method retrieve measurement error rates by running additional circuits containing bit-flips.

The Gate Error Mitigation pattern can be implemented in the form of a gate addition method, such as Zero Noise Extrapolation (ZNE) [54]. Concrete examples of ZNE are FIIM [11], RIIM [11], List Identity Insertion Method (LIIM) [54], or Set Identity Insertion Method (SIIM) [54]. Other methods, e.g., NACL [47], reduce gate error rates based on ML-based circuit learning.

Table I. Real-world occurrences of the introduced patterns

Error Handling Pattern	**Real-world Occurrences**
Error Correction	Laflamme's 5-qubit code, Shor's 9-qubit code, BS codes, Surface codes, CSS codes
Readout Error Mitigation	TPNM, CTMP, M3, DDOT, SIM, AIM, BFA
Gate Error Mitigation	FIIM, RIIM, LIIM, SIIM, NACL

Software-based handling of quantum errors is of utmost importance due to the high error rates of quantum devices in the NISQ era. However, it is expected that error rates will continue to decrease with every new generation of quantum devices, which raises the question of whether error handling will stay of importance in the long term. Due to the fragility of quantum states, it seems unlikely that the occurrence of quantum errors can be avoided entirely. Hence, quantum error handling will remain important and it is rather a question about *which new kinds of error handling techniques* will appear in the future. Error correction is expected to be the most promising long-term solution as it provides the possibility of fault-tolerant quantum computing once the hardware capabilities are sufficient. However, solely focusing on error correction would disregard the state of the current and near-term generation of quantum devices, as their limited capabilities make the application of error correction infeasible and require applying error mitigation techniques.

Furthermore, it is likely that hardware providers will integrate automatic error handling systems in the future. For instance, IBM already integrates the M3 REM method into Qiskit Runtime [55], an environment for running hybrid quantum algorithms close to the quantum device. However, solely relying on such systems limits developers to the options offered by hardware providers, as it is impossible to modify or extend provider APIs, e.g., by optimizing a method's implementation or integrating a newly published error handling method. Therefore, the introduced catalog of quantum error handling patterns, which can also be extended with new patterns documenting other kinds of error handling techniques, is also helpful as a structured guide that facilitates the general understanding of the topic.

V. Related Work

The patterns for quantum error handling introduced in this work extend the existing quantum computing pattern language [13][14][15][16]. Besides these patterns, there are other works that summarize reoccurring concepts in the quantum computing domain [56][57][58]. However, they are not following the pattern concept first introduced by Alexander et al. [12] in the architecture domain. The pattern concept is not restricted to architecture though, it is also widely spread in information technology. Examples are the enterprise integration patterns [59] or the cloud computing patterns [18]. To the best of our knowledge, there have been no other patterns published in the domain of quantum computing.

To facilitate the application of abstract solutions, Falkenthal and Leymann [21] introduced the concept of solution languages. Solution languages provide concrete solutions for specific patterns, e.g., a usable circuit or an implementation of a pattern for a specific language [60]. The concrete solution artifacts are linked to their corresponding patterns and connected to other solutions according to the relations of the pattern language [61]. Thus, concrete circuits and implementations of different methods solving one of the error handling patterns can be provided in a corresponding solution language.

With the continuous increase of transistors and clock speeds in classical computing, reliability has emerged as a critical concern [62]. Hence, novel error correction codes for classical hardware keep getting proposed [63]. It has been shown that concepts from classical error correction can be employed for quantum hardware [64], therefore, the emergence of new classical methods may prove useful to quantum error correction.

In the domain of quantum error handling, there have been multiple works surveying existing methods and explaining the basic concepts of error correction and error mitigation [4][7][37]. However, none of them is guiding users in applying error handling solutions to a given problem at hand. For example, Devitt et al. [7] briefly describe the fundamentals of error correction and then present a variety of methods in detail. Since these detailed method descriptions require a quantum computing background, they are not easy and fast to understand for people that are non-experts and mainly want to get an overview of how they can deal with quantum errors. On the contrary, our work provides easy access to well-structured compact knowledge artifacts, which facilitates a fast understanding of the topic.

VI. Conclusion and Future Work

Although quantum computing advanced rapidly in the last few years, the current generation of quantum devices is still highly error-prone. To achieve meaningful results, quantum software engineers need to understand the limitations and how to minimize the impact of the occurring errors. In this work, we extend the quantum computing patterns, by introducing three new patterns for quantum error handling that shall support quantum software engineers in building and running quantum algorithms successfully on error-prone quantum devices. As quantum computing is an interdisciplinary domain, we first introduce the fundamental terms to establish a common set of vocabulary, explaining the required basic concepts. Then, we present the quantum error handling patterns, explaining and showcasing proven solutions strategies for the prevention and mitigation of different types of quantum errors.

For future work, we plan to incorporate the quantum error handling patterns into PlanQK [65], a platform for sharing knowledge about quantum computing. PlanQK uses the pattern repository Pattern Atlas [66] for the presentation of patterns and contains all currently published quantum computing patterns [67]. By including all quantum computing patterns in such an online platform, a constant re-evaluation and a continuous evolution of the pattern language can be achieved. This also includes an evaluation of the usability of the error handling patterns based on the feedback provided by the community. The evaluation results can then be used to refine the patterns and further improve them. Ensuring a constant re-evaluation is of great importance for the quantum computing pattern language, as the domain is still rapidly evolving, and we expect the emergence of more best practices that we plan to abstract into new patterns that further extend the quantum computing pattern language. Additionally, we plan to implement a quantum computing solution language that will facilitate the application of the patterns presented in this work. The solution language is also planned to be integrated into PlanQK, enabling users to add code snippets and link them to the corresponding pattern.

Acknowledgment

This work was partially funded by the project *SEQUOIA* funded by the Baden-Wuerttemberg Ministry of Economic Affairs, Labour and Tourism and by the BMWK project *PlanQK* (01MK20005N)

References

[1] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[2] Y. Cao *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical Reviews*, vol. 119, no. 19, pp. 10 856–10 915, 2019.

[3] E. Zahedinejad and A. Zaribafiyan, "Combinatorial optimization on gate model quantum computers: A survey," *arXiv preprint arXiv:1708.05294*, 2017.

[4] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the NISQ era," *Quantum Science and Technology*, vol. 5, no. 4, p. 044007, 2020.

[5] M. Salm, J. Barzen, F. Leymann, and B. Weder, "About a Criterion of Successfully Executing a Circuit in the NISQ Era: What $wd \ll 1/\epsilon_{\text{eff}}$ Really Means," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software*, ser. APEQS 2020. ACM, 2020, p. 10–13.

[6] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, pp. R2493–R2496, 1995.

[7] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, no. 7, p. 076001, 2013.

[8] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, "Perfect quantum error correcting code," *Phys. Rev. Lett.*, vol. 77, pp. 198–201, 1996.

[9] R. Matsumoto and M. Hagiwara, "A survey of quantum error correction," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 104, no. 12, pp. 1654–1664, 2021.

[10] S. Bravyi, S. Sheldon, A. Kandala, D. C. Mckay, and J. M. Gambetta, "Mitigating measurement errors in multiqubit experiments," *Physical Review A*, vol. 103, no. 4, p. 042605, 2021.

[11] A. He, B. Nachman, W. A. de Jong, and C. W. Bauer, "Zero-noise extrapolation for quantum-gate error mitigation with identity insertions," *Physical Review A*, vol. 102, no. 1, p. 012426, 2020.

[12] C. Alexander, *A pattern language: towns, buildings, construction*. Oxford university press, 1977.

[13] F. Leymann, "Towards a pattern language for quantum algorithms," in *International Workshop on Quantum Technology and Optimization Problems*. Springer, 2019, pp. 218–230.

[14] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Encoding patterns for quantum algorithms," *IET Quantum Communication*, vol. 2, no. 4, pp. 141–152, 2021.

[15] ——, "Expanding data encoding patterns for quantum algorithms," in *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, 2021, pp. 95–101.

[16] M. Weigold, J. Barzen, F. Leymann, and D. Vietz, "Patterns for hybrid quantum algorithms," in *Symposium and Summer School on Service-Oriented Computing*. Springer, 2021, pp. 34–51.

[17] J. O. Coplien and A. W. O. Alexander, "Software patterns," 1996.

[18] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, *Cloud computing patterns: fundamentals to design, build, and manage cloud applications*. Springer, 2014.

[19] E. Gamma, R. Helm, R. Johnson, J. Vlissides, and D. Patterns, *Elements of reusable object-oriented software*. Addison-Wesley Reading, Massachusetts, 1995, vol. 99.

[20] V. Yussupov, J. Soldani, U. Breitenbücher, A. Brogi, and F. Leymann, "From serverful to serverless: A spectrum of patterns for hosting application components." in *CLOSER*, 2021, pp. 268–279.

[21] M. Falkenthal and F. Leymann, "Easing pattern application by means of solution languages," in *Proceedings of the 9th International Conference on Pervasive Patterns and Applications*, 2017, pp. 58–64.

[22] IBM, "Overview of ibm quantum computing systems," https://www.ibm.com/quantum-computing/systems/ [accessed: 2022.01.24].

[23] Amazon, "Overview of Amazon Braket Pricing," https://aws.amazon.com/de/braket/pricing/ [accessed: 2022.01.24].

[24] A. Peruzzo *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, pp. 1–7, 2014.

[25] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[26] M. Cerezo *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, pp. 1–20, 2021.

[27] S. Sivarajah *et al.*, "t| ket⟩: a retargetable compiler for NISQ devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, 2020.

[28] S. S. Tannu and M. K. Qureshi, "Mitigating measurement errors in quantum computers by exploiting state-dependent bias," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2019, p. 279–290.

[29] M. Salm *et al.*, "The NISQ Analyzer: Automating the selection of quantum computers for quantum algorithms," in *Symposium and Summer School on Service-Oriented Computing*. Springer, 2020, pp. 66–85.

[30] B. Weder, J. Barzen, F. Leymann, and D. Vietz, "Quantum software development lifecycle," in *Quantum Software Engineering, arXiv preprint arXiv:2106.05800*. Springer, 2022.

[31] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[32] J. Chiaverini *et al.*, "Realization of quantum error correction," *Nature*, vol. 432, no. 7017, pp. 602–605, 2004.

[33] W. K. Wootters and W. H. Zurek, "The no-cloning theorem," *Physics Today*, vol. 62, no. 2, pp. 76–77, 2009.

[34] W. Cai, Y. Ma, W. Wang, C.-L. Zou, and L. Sun, "Bosonic quantum error correction codes in superconducting quantum circuits," *Fundamental Research*, vol. 1, no. 1, pp. 50–67, 2021.

[35] D. Gottesman, *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[36] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, no. 3, pp. 226–245, 2019.

[37] B. Nachman, M. Urbanek, W. A. de Jong, and C. W. Bauer, "Unfolding quantum computer readout noise," *npj Quantum Information*, vol. 6, no. 1, pp. 1–7, 2020.

[38] F. B. Maciejewski, Z. Zimborás, and M. Oszmaniec, "Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography," *Quantum*, vol. 4, p. 257, 2020.

[39] P. D. Nation, H. Kang, N. Sundaresan, and J. M. Gambetta, "Scalable mitigation of measurement errors on quantum computers," *PRX Quantum*, vol. 2, no. 4, p. 040326, 2021.

[40] A. W. Smith, K. E. Khosla, C. N. Self, and M. Kim, "Qubit readout error mitigation with bit-flip averaging," *arXiv preprint arXiv:2106.05800*, vol. 7, no. 47, p. eabi8009, 2021.

[41] M. Streif, M. Leib, F. Wudarski, E. Rieffel, and Z. Wang, "Quantum algorithms with local particle-number conservation: Noise effects and error correction," *Physical Review A*, vol. 103, no. 4, p. 042412, 2021.

[42] S. Seo, J. Seong, and J. Bae, "Mitigation of crosstalk errors in a quantum measurement and its applications," 2021.

[43] T. Alexander *et al.*, "Qiskit pulse: programming quantum computers through the cloud with pulses," *Quantum Science and Technology*, vol. 5, no. 4, p. 044006, 2020.

[44] Rigetti, "Quil-t: an extension for the pulse-level control of quantum programs," https://pyquil-docs.rigetti.com/en/stable/quilt.html [accessed: 2022.01.24].

[45] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Physical review letters*, vol. 119, no. 18, p. 180509, 2017.

[46] R. Harper and S. T. Flammia, "Fault-tolerant logical gates in the ibm quantum experience," *Phys. Rev. Lett.*, vol. 122, p. 080504, 2019.

[47] L. Cincio, K. Rudinger, M. Sarovar, and P. J. Coles, "Machine learning of noise-resilient quantum circuits," *PRX Quantum*, vol. 2, no. 1, p. 010324, 2021.

[48] A. R. Carvalho, H. Ball, M. J. Biercuk, M. R. Hush, and F. Thomsen, "Error-robust quantum logic optimization using a cloud quantum computer interface," *Physical Review Applied*, vol. 15, no. 6, p. 064054, 2021.

[49] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, "Digital zero noise extrapolation for quantum error mitigation," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2020, pp. 306–316.

[50] D. Bacon, "Operator quantum error-correcting subsystems for self-correcting quantum memories," *Phys. Rev. A*, vol. 73, p. 012340, Jan 2006.

[51] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, vol. 54, pp. 1098–1105, Aug 1996.

[52] A. Steane, "Multiple-particle interference and quantum error correction," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 452, no. 1954, pp. 2551–2577, 1996.

[53] F. B. Maciejewski, F. Baccari, Z. Zimborás, and M. Oszmaniec, "Modeling and mitigation of realistic readout noise with applications to the quantum approximate optimization algorithm," *arXiv preprint arXiv:2101.02331*, 2021.

[54] C. Bauer, A. He, W. A. de Jong, B. Nachman, and V. R. Pascuzzi, "Computationally efficient zero noise extrapolation for quantum gate error mitigation," *arXiv preprint arXiv:2110.13338*, 2021.

[55] IBM, "Overview of qiskit runtime," https://github.com/Qiskit-Partners/qiskit-runtime [accessed: 2022.01.24].

[56] A. Gilliam *et al.*, "Foundational patterns for efficient quantum computing," *arXiv preprint arXiv:1907.11513*, 2019.

[57] Y. Huang and M. Martonosi, "Statistical assertions for validating patterns and finding bugs in quantum programs," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 541–553.

[58] S. Perdrix, "Quantum patterns and types for entanglement and separability," *Electronic Notes in Theoretical Computer Science*, vol. 170, pp. 125–138, 2007.

[59] G. Hohpe and B. Woolf, "Enterprise integration patterns," in *9th conference on pattern language of programs*, 2002, pp. 1–9.

[60] M. Falkenthal, J. Barzen, U. Breitenbücher, C. Fehling, and F. Leymann, "Efficient pattern application: Validating the concept of solution implementations in different domains," vol. 7, no. 3&4. Xpert Publishing Services (XPS), 2014, pp. 710–726.

[61] M. Falkenthal *et al.*, "Leveraging pattern application via pattern refinement," in *Proceedings of the International Conference on Pursuit of Pattern Languages for Societal Change (PURPLSOC)*, 2016, pp. 38–61.

[62] M. M. Hafidhi and E. Boutillon, "Hardware error correction using local syndromes," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2017, pp. 1–6.

[63] L.-J. Saiz-Adalid *et al.*, "Reducing the overhead of bch codes: New double error correction codes," *Electronics*, vol. 9, no. 11, p. 1897, 2020.

[64] D. MacKay, G. Mitchison, and P. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2315–2330, 2004.

[65] PlanQK, "Planqk - the platform for quantum-supported artificial intelligence," https://platform.planqk.de/ [accessed: 2022.01.24].

[66] F. Leymann and J. Barzen, *Pattern Atlas*. Springer International Publishing, 2021, pp. 67–76.

[67] PlanQK, "Planqk - integration of the patternrepository pattern atlas," https://patterns.platform.planqk.de/pattern-languages [accessed: 2022.01.24].