PATTERNS 2025

The Seventeenth International Conferences on Pervasive Patterns and Applications

ISBN: 978-1-68558-263-0

April 6 - 10, 2025

Valencia, Spain

PATTERNS 2025 Editors

Herwig Mannaert, University of Antwerp, Belgium

PATTERNS 2025

Forward

The Seventeenth International Conferences on Pervasive Patterns and Applications (PATTERNS 2025), held on April 6 – 10, 2025, continued a series of events targeting the application of advanced patterns, at-large. In addition to support for patterns and pattern processing, special categories of patterns covering ubiquity, software, security, communications, discovery and decision were considered. It is believed that patterns play an important role on cognition, automation, and service computation and orchestration areas. Antipatterns come as a normal output as needed lessons learned.

The conference had the following tracks:

- Patterns basics
- Patterns at work
- Discovery and decision patterns
- Medical and facial image patterns
- Tracking human patterns

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the PATTERNS 2025 technical program committee, as well as the numerous reviewers. The creation of a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to PATTERNS 2025. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the PATTERNS 2025 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope PATTERNS 2025 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of pervasive patterns and applications. We also hope that Valencia provided a pleasant environment during the conference and everyone saved some time to enjoy this beautiful city.

PATTERNS 2025 Steering Committee

Herwig Manaert, University of Antwerp, Belgium
Wladyslaw Homenda, Warsaw University of Technology, Poland
Yuji Iwahori, Chubu University, Japan
Alexander Mirnig, University of Salzburg, Austria
George A. Papakostas, International Hellenic University – Kavala, Greece

PATTERNS 2025 Publicity Chair

Francisco Javier Díaz Blasco, Universitat Politècnica de València, Spain
Ali Ahmad, Universitat Politècnica de València, Spain
José Miguel Jiménez, Universitat Politècnica de València, Spain
Sandra Viciano Tudela, Universitat Politècnica de València, Spain

PATTERNS 2025

Committee

PATTERNS 2025 Steering Committee

Herwig Manaert, University of Antwerp, Belgium
Wladyslaw Homenda, Warsaw University of Technology, Poland
Yuji Iwahori, Chubu University, Japan
Alexander Mirnig, University of Salzburg, Austria
George A. Papakostas, International Hellenic University – Kavala, Greece

PATTERNS 2025 Publicity Chair

Francisco Javier Díaz Blasco, Universitat Politècnica de València, Spain
Ali Ahmad, Universitat Politècnica de València, Spain
José Miguel Jiménez, Universitat Politècnica de València, Spain
Sandra Viciano Tudela, Universitat Politècnica de València, Spain

PATTERNS 2025 Technical Program Committee

Andrea F. Abate, University of Salerno, Italy
Akshay Agarwal, IIIT Delhi, India
Carlos Alexandre Ferreira, INESC TEC, Portugal
Vijayan K. Asari, University of Dayton, USA
Danilo Avola, Sapienza University of Rome, Italy
Johanna Barzen, University of Stuttgart, Germany
Frederik Simon Bäumer, Bielefeld University of Applied Sciences, Germany
Martin Beisel, University of Stuttgart, Germany
Nadjia Benblidia, Saad Dahlab University - Blida1, Algeria
Anna Berlino, Consultant in Tourism Sciences and Valorization of Cultural and Tourism Systems, Italy
Fatma Bouhlel, University of Sfax, Tunisia
Uwe Breitenbücher, Reutlingen University, Germany
Alceu S. Britto, Pontifical Catholic University of Paranā (PUCPR), Brazil
Eliot Bytyçi, University of Prishtina "Hasan Prishtina", Kosovo
Isaac Caicedo-Castro, University of Córdoba, Colombia
Simone Cammarasana, CNR-IMATI, Genova, Italy
David Cárdenas-Peña, Universidad Tecnológica de Pereira, Colombia
Bidyut B. Chaudhuri, Indian Statistical Institute, India
Sneha Chaudhari, AI Organization | LinkedIn, USA
Diego Collazos, Universidad Nacional de Colombia sede Manizales, Colombia
Sergio Cruces, University of Seville, Spain
Mohamed Daoudi, Institut Mines-Telecom / Telecom Lille, France
Jacqueline Daykin, King's College London, UK / Aberystwyth University, Wales & Mauritius

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

LLM-Based Design Pattern Detection

Christian Schindler and Andreas Rausch
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
e-mail: {christian.schindler | andreas.rausch}@tu-clausthal.de

***Abstract*—Detecting design pattern instances in unfamiliar codebases remains a challenging yet essential task for improving software quality and maintainability. Traditional static analysis tools often struggle with the complexity, variability, and lack of explicit annotations that characterize real-world pattern implementations. In this paper, we present a novel approach leveraging Large Language Models (LLMs) to automatically identify design pattern instances across diverse codebases. Our method focuses on recognizing the roles classes play within the pattern instances. By providing clearer insights into software structure and intent, this research aims to support developers, improve comprehension, and streamline tasks such as refactoring, maintenance, and adherence to best practices.**

Keywords-Design Pattern detection; Large Language Model.

I. Introduction

Identifying design pattern instances in code is a valuable goal as it enables a deeper understanding of the structural and behavioral principles underlying software systems. By uncovering these patterns, developers and other stakeholders can gain insights into code quality, maintainability, and adherence to best practices, even in unfamiliar code bases [1]. Automating this process can significantly reduce the time and effort required for code comprehension, facilitate knowledge transfer among teams, and improve software evolution and refactoring efforts. Furthermore, it can aid in identifying reusable components, fostering consistency, and enhancing the overall robustness of software design.

It is a challenging task due to several factors. Design patterns are often implemented with significant variations tailored to specific use cases, making consistent recognition difficult [2]. Developers frequently deviate from canonical implementations or introduce domain-specific adaptations, complicating detection efforts [3]. Additionally, design patterns are typically embedded implicitly within the code's structure and behavior, rather than being explicitly annotated or identifiable by a set of keywords, requiring a deep contextual understanding of the code, its dependencies, and its intent. Furthermore, applying pattern detection techniques to large, complex, and poorly documented code bases poses scalability challenges, as the sheer volume of code and intertwined dependencies can overwhelm traditional approaches. These factors collectively highlight the complexity of automating design pattern identification in diverse and unfamiliar code bases.

We have defined the following Research Questions (RQs). RQ1: How can LLMs be leveraged to automatically detect and annotate design pattern instances in software codebases? RQ2: How good can LLMs detect and annotate design pattern instances in software codebases? RQ3: What are challenges/limitations faced by LLMs in identifying design pattern instances in code bases, and how can these be addressed?

This paper is organized as follows. In Section 2, we motivate the work and define the main research task. Section 3 surveys related work, while Section 4 details the experimental setup and describes the dataset. Section 5 presents the experimental results. Finally, Section 6 concludes the paper and outlines directions for future research.

II. Motivating example

The idea of a design pattern in software engineering is to provide a reusable, general solution to a commonly occurring problem within a given context in software design. Design patterns encapsulate best practices and proven strategies for solving these problems, offering a structured approach to building robust, maintainable, and flexible software systems. The design pattern is described on a conceptual level, defining roles and their specific responsibilities within the pattern to achieve its intended design purpose.

A design pattern instance refers to the concrete implementation of a design pattern within a specific piece of software. In this context-specific realization, the roles defined by the design pattern are embodied by individual components, such as classes, objects, or packages, which collaboratively fulfill the pattern's intended structure and behavior.

The task we want to work on is to localize such design pattern instances in code. We want to detect the design pattern applied with the respective pairs of components and their roles.

III. Related Work

A. Design Pattern Detection

Detecting design pattern instances in existing code is a desirable task in later stages of a software system's lifecycle, particularly for maintenance purposes and to facilitate the onboarding of new developers. Various approaches have been proposed to address this challenge.

Matching-based approaches are common, such as those that rely on similarity scores [4] or graph structures [5] [6]. They have several limitations. Firstly, they often require a high degree of structural similarity between the detected instances and the target pattern. For example, they may mandate an identical number of roles or a specific implementation style. Secondly, the computational cost of the detection process can be significant, depending on the chosen matching strategy.

 ISBN: 978-1-68558-263-0

Furthermore, multi-stage approaches have been explored. These often involve a learning phase [7] [8] or a pattern definition phase [9] [10] prior to the actual detection of design pattern instances. Learning-based approaches in the first stage are limited by the requirement of a substantial amount of annotated training data. This data must encompass the necessary diversity of design pattern instances for the patterns of interest.

(Semi-)formal definition of design pattern structures also faces limitations. These include the expressiveness of the chosen language, the mapping of abstract concepts to language-specific constructs, and the inherent difficulty of defining patterns at this level of abstraction. The quality of this definition significantly impacts the subsequent detection phase and the accuracy of the results.

Several approaches focus on classifying source code as design pattern instances based on fixed-length inputs, such as a single class [11] [12] [13]. These methods typically work well for small or well-defined code segments where the scope is narrowly confined. However, they may fail to capture the broader context of how multiple classes or modules interact, leading to fragmented or incomplete representations of the software. In an attempt to handle larger portions of code, some methods aggregate metrics over entire modules or files—such as by summing values into a fixed-size feature vector [14]. While this strategy can reduce dimensionality and simplify processing, it tends to obscure important semantic and structural details because numerous code properties get merged into a single set of features. FeatRacer [15] addresses feature location by combining manual annotations with automated suggestions. It uses machine learning to predict missing feature recordings and guide developers, resulting in more accurate and complete feature information.

B. Large Language Models

LLMs are advanced language models that excel at understanding and processing human language [16]. They handle tasks with minimal examples, especially involving text and data tables [17]. With larger models the performance increases [18]. Generative Pre-trained Transformers (GPTs) are powerful transformer-based models with broad applications (e.g., in education, healthcare) and challenges like high computational demands, interpretability, and ethical issues [19]. Extensively studied in computer science, GPTs relates to AI, language processing, and deep learning, as Cano [20] noted. It also aids engineering design by generating novel ideas, as Gill [21] and Zhu [22] discovered.

IV. Experiment

A. Data Set of Design Pattern Instances

The data used in this work was originally collected and published by [23], and has since been adopted in various subsequent studies for benchmarking. In addition to annotating design pattern instances, the authors also provided specific versions of the corresponding source code. In this paper, we focus on instances of the *Composite* design pattern.

B. Experimental setup

The experimental setup, depicted in Figure 1, is discussed in this section.

1) Data Preparation: During this phase, we examined the available annotated design pattern instances to ensure their suitability for analysis. We identified and filtered out instances where not all associated source code files were available, as incomplete datasets could compromise the reliability of subsequent steps. Furthermore, we implemented a basic yet essential data cleaning procedure to improve the consistency of the annotations. This involved trimming unnecessary blank spaces in the Extensible Markup Language (XML) files containing the ground truth annotations, ensuring that formatting inconsistencies did not introduce errors or ambiguities.

2) Prompt Preparation: Figure 1 illustrates the structure of the prompts. The first prompt includes a general setup section that provides context for the LLM, clearly defined instructions, additional remarks, and a sample consisting of a source code snippet alongside the design pattern instance annotation.

In addition to following the prompt structure, we need to consider limiting factors in preparing the prompts A first restriction for the prompts is the available token size of the LLM. We used ChatGPT4 and the predecessor ChatGPT3.5, both with a limit of 128k tokens.

Because this represents the upper bound, our interaction with the LLM must include both prompts we provide and the expected answer, which will be much smaller than the input portion. To determine which classes to include in the provided example snippet and in the snippet for which the design pattern needs to be identified, we first identify the common root package shared by all classes participating in the ground truth annotation. For instance, the classes *a.b.c.d.ClassA* and *a.b.e.f.ClassB* share the common root package *a.b.*.

Table I provides a comprehensive overview of all remaining examples of the Composite design pattern after the Data Preparation step. For each example, it includes details about the originating project, the identified common root package, and the number of classes that (i) actively participate in the design pattern and (ii) belong to the common root package. To further optimize the input size, we have removed all comments from the classes. This reduction ensures that only essential code components remain, making the input more concise and focused for subsequent analysis and processing.

The output of this step is a collection of prompt messages. Each prompt includes a dedicated example containing a source code snippet paired with its ground truth annotation for the design pattern instance, as well as a second source code snippet from a different Java project. All permutations of the two source code snippets are considered, provided their combined token count falls within the token limit of the LLM. The order of the pairings is significant, as one snippet serves as the provided example while the other is the source code sample to be analyzed and annotated.

3) Utilization of the LLM: The actual prediction task is performed using ChatGPT3.5 and 4, though other LLMs could also be used due to their similar handling and functionality.

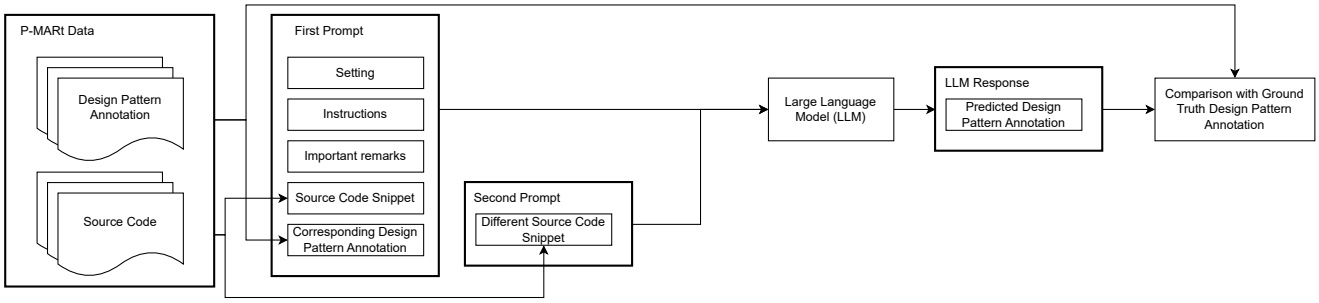


Figure 1. Schematic Overview of the Conducted Experiment.

TABLE I. INSIGHT INTO THE DESIGN PATTERN INSTANCES.

Instance	*Project*	*Common Root Package*	*# Classes Participating in Pattern instance*	*# Classes in Root Package*
4	QuickUML 2001		17	217
65	JUnit v3.7	junit	39	94
75	JHotDraw v5.1	CH.ifa.draw	35	155
98	MapperXML v1.9.7	com.taursys	29	234
129	PMD v1.8	net.sourceforge.pmd.ast	3	108
143	Software architecture design patterns in Java	src.COMPOSITE	5	5

Each prompt is presented to the LLM, and the generated output, which is expected to be in a valid XML format, is saved. The output is stored alongside the respective example and the design pattern expected to be found, ensuring traceability and alignment for subsequent analysis.

4) Post Processing: We analyzed the responses provided by the LLM and systematically extracted relevant XML annotations whenever they were available as the output of the model. In cases where the LLM did not provide any annotations, it responded with a clear answer thast no design pattern instance has been found. This allowed us to differentiate between cases where design patterns were explicitly recognized and cases where their absence was confirmed based on the LLM's output.

V. RESULTS

All prompts, LLM-generated responses, and the ground truth are available[1]. Table II presents a confusion matrix (left hand side) illustrating the roles associated with the *Composite* design pattern. Each column of the matrix corresponds to a distinct role, as well as to the additional categories *No Role* and *Hallucinated Class*. The rows follow a similar structure, with the key difference being the use of *Hallucinated Role* in place of *Hallucinated Class*. Within each cell, the matrix reports the total number of occurrences in which the LLM predicted a particular role (column) given a specific ground-truth role (row). For example, the cell at the intersection of the *Leaf* row and the *Composite* column indicates that, on four separate occasions, a class that was actually labeled as *Leaf* was misclassified by the LLM as *Composite*.

The right hand side of the table display the total number of classifications made for each role and the corresponding Precision with and without the hallucinated predictions of the LLM. Likewise, the bottom two rows (twice) of the table show the total number of ground-truth occurrences of each role, along with the corresponding Recall. The cells at the intersection of these totals represents the overall number.

Hallucination, a well-documented behavior in LLMs, involves blending prompt elements or generating information unsupported by any source. Our experiments also exhibited this phenomenon. In particular, we observed the model inventing classes not present in the code snippets—an occurrence we term hallucinated classes. Beyond that, we noted a second form of hallucination: the creation of new roles not found in any of the ground-truth examples. Both forms of hallucination are relatively straightforward to detect, as we have direct access to the complete set of valid labels and the classes embedded in the source code snippets. This comprehensive dataset enables a thorough verification process: we can systematically compare the model's predictions against the known ground-truth labels and source code elements. By doing so, it becomes evident when the model invents new classes that never appear in the provided snippets, or assigns roles not included in any of the reference examples.

We also report the results for individual prompts in Table IV and Table V. All roles—except for *No Role*— combined are the positive prediction class. We define the standard confusion matrix terms as follows: True Positive (TP): The model correctly identifies that a given source code element serves a specific role in a design pattern instance. True Negative (TN): The model correctly identifies that a given source code element does not participate in any role of the design pattern instance. False Positive (FP): The model incorrectly classifies a source code element as fulfilling a particular role, even though it does not. False Negative (FN): The model incorrectly concludes that a source code element does not fulfill any role when, in fact,

[1]https://github.com/schindlerc/LLM-Based-Design-Pattern-Detection

TABLE II. CONFUSION MATRIX CHATGPT 3.5.

							With Hallucination		Without Hallucination	
	Hallucinated Class	Client	Component	Composite	Leaf	No Role	Classification Overall	Precision	Classification Overall	Precision
Hallucinated Role	0	0	1	0	0	3	4	-	-	-
Client	38	5	1	0	2	24	70	.071	32	.156
Component	0	0	8	1	0	1	10	.800	10	.800
Composite	1	0	0	4	12	10	27	.148	26	.154
Leaf	13	0	0	4	17	32	66	.258	53	.321
No Role	0	23	5	16	185	1064	1293	.823	1293	.823
Truth overall	52	28	15	25	216	1134	1470			
Recall	-	.179	.533	.160	.008	.938				
Truth overall	-	28	14	25	216	1131			1414	
Recall	-	.179	.571	.160	.008	.941				

TABLE III. CONFUSION MATRIX GPT4.

	No Role	Client	Component	Composite	Leaf	Classification Overall	Precision
Client	13	9	1	3	8	34	.265
Component	1	0	10	0	0	11	.910
Composite	5	0	0	12	12	29	.414
Leaf	34	2	0	1	45	82	.549
No Role	1079	17	4	11	150	1261	.856
Truth overall	1132	28	15	27	215	1417	
Recall	.953	.333	.833	.522	.209		

it does have a defined role according to the ground truth.

That implies that the Precision, Recall, and F1 metrics reported focus solely on assessing the model's ability to correctly identify the roles within the design pattern instances. All classes from the code snippets that are not part of the design pattern instance serve as the negative (prediction) class, ensuring that the evaluation focuses on the roles of the design pattern. In evaluating the performance of classification models, Precision, Recall, and F1 Score are key metrics that provide insight into the model's accuracy in distinguishing between the classes. These metrics are particularly important in imbalanced datasets where a models overall accuracy might be misleading.

In the context of FP, a misalignment between the intended usage of a design pattern and its implementation can negatively impact software engineering projects, resulting in bugs, unexpected behavior, and error propagation throughout the system.

Table IV shows the different prediction runs with ChatGPT 3.5 as a row each, with the following information. *Run* is a running number to refer to the individual prediction runs with *TP, TN, FP*, and *FN* predictions, alongside the bespoken metrics. On the right side we reported the same metrics after removed the hallucination introduced by the LLM. Hallucination occurred in three runs (2, 3, and 4). Removal of the hallucination lead to increased Precision, F1 Score and Accuracy in those runs. Four runs (i.e., 11, 12, 13, and 14) did not detect a design pattern annotation but instead responded with the no design pattern found response.

Table V lists the prediction runs with ChatGPT 4, the noticeable difference to Table IV is, that no hallucination occurred. In a consequence, for each run their is only one set of metrics to be reported. In addition ChatGPT 4 has been better in 7 out of 14 runs for Precision and Accuracy, in 8 cases for F1 Score and in 9 cases for Recall.

Table VI provides additional details on each of the 14 runs and includes the corresponding design pattern instance IDs for both the example and the target, referencing the IDs in Table I. In this context, the example is the annotated instance, while the target is the instance for which only source code was provided via prompts. The subsequent columns show the number of role annotations in the ground truth (for both Example and Target) and those predicted by the LLMs ChatGPT 3.5 and 4.

We noticed that in all cases where the LLMs failed to identify a design pattern annotation, the same example (143) has been used. A notable aspect of this example is that the source code snippet includes only those classes that participate in the design pattern (see Table I), with no additional classes. It appears that providing the snippet within a broader implementation context helps the LLM better distinguish which classes are relevant to a design pattern instance.

Examining the confusion matrix shows that Precision and Recall vary considerably across different roles. The highest scores for both Precision and Recall occur for the role *Component* (keeping the *No Role* label out of scope), likely because each design pattern instance has exactly one class labeled with this role, and every successful prediction also identified exactly one such class. By contrast, performance for the other roles is poorer. As shown in Table VI, the number of classes labeled differs widely among examples. In the case of *Client*, for instance, some instances lack this role entirely.

To address RQ1, we designed two prompts for the task. The first prompt provides essential context: it describes the setting for the LLM, outlines its assumed skill set, and provides explicit instructions for problem-solving and important remarks. It also includes an example of an annotated design pattern and a corresponding code snippet. The second prompt supplies the target code snippet for which the model should output

TABLE IV. RESULTS FOR EACH PREDICTION RUN WITH CHATGPT 3.5

	With Hallucination								Without Hallucination							
Run	TP	FP	FN	TN	Precision (P)	Recall (R)	F1	Accuracy (A)	TP	FP	FN	TN	P	R	F1	A
1	4	5	36	53	.444	.100	.163	.582	4	5	36	53	.444	.100	.163	.582
2	12	17	23	117	.414	.343	.375	.763	12	**6**	23	117	**.667**	.343	**.453**	**.816**
3	3	77	37	39	.037	.075	.050	.269	3	**33**	37	39	**.083**	.075	**.079**	**.375**
4	2	2	38	54	.500	.050	.091	.583	2	**1**	38	54	**.667**	.050	**.093**	**.589**
5	1	29	16	171	.033	.059	.043	.793	1	29	16	171	.033	.059	.043	.793
6	1	3	4	0	.250	.200	.222	.125	1	3	4	0	.250	.200	.222	.125
7	3	4	2	0	.429	.600	.500	.333	3	4	2	0	.429	.600	.500	.333
8	3	1	2	0	.750	.600	.667	.500	3	1	2	0	.750	.600	.667	.500
9	3	1	2	0	.750	.600	.667	.500	3	1	2	0	.750	.600	.667	.500
10	2	3	3	0	.400	.400	.400	.250	2	3	3	0	.400	.400	.400	.250
11	0	0	17	200	0	0	0	.922	0	0	17	200	0	0	0	.922
12	0	0	3	105	0	0	0	.972	0	0	3	105	0	0	0	.972
13	0	0	29	205	0	0	0	.876	0	0	29	205	0	0	0	.876
14	0	0	35	120	0	0	0	.774	0	0	35	120	0	0	0	.774

TABLE V. RESULTS FOR EACH PREDICTION RUN WITH CHATGPT4

Run	TP	FP	FN	TN	Precision	Recall	F1	Accuracy
1	6	3	34	54	.667	.150	.245	.619
2	21	32	14	98	.396	.600	.477	.721
3	8	4	32	53	.667	.200	.308	.629
4	0	0	39	55	0	0	0	.585
5	0	0	17	200	0	0	0	.922
6	2	1	3	0	.667	.400	.500	.333
7	3	1	2	0	.750	.600	.667	.500
8	3	1	2	0	.750	.600	.667	.500
9	3	1	2	0	.750	.600	.667	.500
10	2	1	3	0	.667	.400	.500	.333
11	11	0	6	200	1.000	.647	.786	.972
12	0	0	3	105	0	0	0	.972
13	0	11	29	203	0	0	0	.835
14	17	15	18	111	.531	.486	.507	.795

the design pattern annotation. By presenting the context and an example before showing the target snippet, we ensure the LLM is both well-prepared and guided by a concrete sample.

To address RQ2, we analyzed the experimental results, which revealed varying levels of quality depending on the roles within the design pattern. Furthermore ChaGPT 4 performed better than ChatGPT 3.5, by (i) finding more annotations and (ii) having more correct annotations. Additionally, we observed substantial variability in quality across different runs, influenced by the pairing of the example code with unseen target code and the expected annotations.

In RQ3, we investigated the challenges and limitations of using an LLM-based approach for detecting design pattern instances. Our findings demonstrate that LLMs, such as ChatGPT 3.5 and 4 in our case, are capable of processing prompts containing code snippets spanning hundreds of Java classes and accurately retrieving classes along with their correct roles. However, this approach faces several limitations.

One significant constraint is the token limit of current LLMs, which restricts scalability. In our experiments, we could only include a single example of the design pattern for a given prediction in such detail, as multiple examples would exceed the context window. Additionally, some pairings of examples and target code snippets had to be excluded because their combined token count exceeded the allowable limit.

Another challenge lies in the heterogeneity of the design pattern instances, as they vary in the number of roles and the specific roles annotated in the examples, adding complexity to the task and potentially affecting the model's performance.

If the role expected to be predicted by the LLM is not present in the provided example, it becomes an unseen role, making it challenging for the models to annotate accurately.

To address the maxing out of the token size limitation of current LLMs, an effective strategy could involve reducing the size of the code snippets provided in each prompt. By pruning the code snippets—such as focusing more on the relevant sections or removing parts that are less critical for identifying the design pattern. Presenting multiple examples could help the model better generalize across varying instances of the design pattern and potentially improve its prediction accuracy.

For instance, pruning could involve removing sections of code unrelated to the design pattern being analyzed or adjusting the balance between classes directly involved in the design pattern instances and those present in the source project but not participating in the design pattern instance. This approach would allow for a richer variety of training examples without exceeding the token limit, thereby balancing the trade-off between example diversity and code snippet size.

VI. CONCLUSION AND FUTURE WORK

We have demonstrated an approach for identifying design pattern instances in large codebases comprising over 200 classes presented simultaneously, using only a single example of the design pattern. This approach operates directly on the source code without requiring modifications like abstraction or feature extraction. Our results show that the model can partially retrieve design patterns with varying levels of completeness and accuracy.

However, as discussed in the results section of the paper, the approach is not perfect, as it handles different roles with varying degrees of success. Moreover, the experiment we conducted was quite limited: it tested only one design pattern with two LLMs. Consequently, results for other design patterns may differ from those reported here, particularly because certain patterns could be either easier or more difficult to detect.

TABLE VI. COMPARISON OF THE AMOUNT OF ROLES PER EXAMPLE AND RUN.

Run	Design Pattern Instance ID		Composite				Component				Leaf				Client			
	Example (E)	Target (T)	E	T	GPT4	GPT3.5	E	T	4	3.5	E	T	4	3.5	E	T	4	3.5
1	4	65	1	2	1	1	1	1	1	1	14	37	6	6	1	0	1	1
2	65	75	2	5	4	2	1	1	1	1	37	21	29	22	0	8	19	0
3	75	65	5	2	1	10	1	1	1	1	21	37	7	17	8	0	3	52
4	143	129	1	1	-	1	1	1	-	1	2	1	-	1	1	0	-	1
5	143	98	1	1	-	5	1	1	-	1	2	22	-	15	1	5	-	9
6	129	143	1	1	1	2	1	1	1	1	1	2	1	1	0	1	0	0
7	98	143	1	1	1	1	1	1	1	1	22	2	1	1	5	1	1	4
8	65	143	2	1	1	1	1	1	1	1	37	2	1	1	0	1	1	1
9	4	143	1	1	1	1	1	1	1	1	14	2	1	1	1	1	1	1
10	75	143	5	1	1	2	1	1	1	1	21	2	1	1	8	1	0	1
11	143	4	1	1	1	-	1	1	1	-	2	14	8	-	1	1	1	-
12	143	129	1	1	-	-	1	1	-	-	2	14	-	-	1	0	-	-
13	143	98	1	1	5	-	1	1	1	-	2	22	2	-	1	5	3	-
14	143	75	1	5	5	-	1	1	1	-	2	21	25	-	1	8	1	-

However, the presented approach can be consistently applied to other design patterns without modification. In addition, the experiment was conducted as a one-shot test, providing only a single example. It would therefore be valuable to investigate how providing multiple examples might influence the model's predictive performance.

Potential directions for future work in this domain may include fine-tuning LLMs before conducting similar experiments, refining prompt design, and adopting hybrid approaches that integrate LLMs with logical reasoning or static and dynamic analysis.

REFERENCES

[1] O. Kaczor, Y.-G. Guéhéneuc, and S. Hamel, "Identification of design motifs with pattern matching algorithms," *Information and Software Technology*, vol. 52, no. 2, pp. 152–168, 2010.

[2] H. Yarahmadi and S. M. H. Hasheminejad, "Design pattern detection approaches: A systematic review of the literature," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5789–5846, Dec. 2020.

[3] F. A. Fontana, A. Caracciolo, and M. Zanoni, "Dpb: A benchmark for design pattern detection tools," in *2012 16th European Conference on Software Maintenance and Reengineering*, 2012, pp. 235–244.

[4] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S. T. Halkidis, "Design pattern detection using similarity scoring," *IEEE transactions on software engineering*, vol. 32, no. 11, pp. 896–909, 2006.

[5] B. B. Mayvan and A. Rasoolzadegan, "Design pattern detection based on the graph theory," *Knowledge-Based Systems*, vol. 120, pp. 211–225, 2017.

[6] R. Singh Rao and M. Gupta, "Design pattern detection by greedy algorithm using inexact graph matching," *International Journal Of Engineering And Computer Science*, vol. 2, no. 10, pp. 3658–3664, 2013.

[7] N. Bozorgvar, A. Rasoolzadegan, and A. Harati, "Probabilistic detection of gof design patterns," *The Journal of Supercomputing*, vol. 79, no. 2, pp. 1654–1682, 2023.

[8] R. Barbudo, A. Ramírez, F. Servant, and J. R. Romero, "Geml: A grammar-based evolutionary machine learning approach for design-pattern detection," *Journal of Systems and Software*, vol. 175, p. 110 919, 2021.

[9] G. Rasool and P. Mäder, "Flexible design pattern detection based on feature types," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, IEEE, 2011, pp. 243–252.

[10] H. Thaller, L. Linsbauer, and A. Egyed, "Feature maps: A comprehensible software representation for design pattern detection," in *2019 IEEE 26th international conference on software analysis, evolution and reengineering (SANER)*, IEEE, 2019, pp. 207–217.

[11] N. Nazar, A. Aleti, and Y. Zheng, "Feature-based software design pattern detection," *Journal of Systems and Software*, vol. 185, p. 111 179, 2022.

[12] A. Nacef, S. Bahroun, A. Khalfallah, and S. B. Ahmed, "Features and supervised machine learning-based method for singleton design pattern variants detection," in *Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2023)*, 2023, pp. 226–237.

[13] R. Mzid, I. Rezgui, and T. Ziadi, "Attention-based method for design pattern detection," in *European Conference on Software Architecture*, Springer, 2024, pp. 86–101.

[14] S. Komolov, G. Dlamini, S. Megha, and M. Mazzara, "Towards predicting architectural design patterns: A machine learning approach," *Computers*, vol. 11, no. 10, p. 151, 2022.

[15] M. Mukelabai, K. Hermann, T. Berger, and J.-P. Steghöfer, "FeatRacer: Locating features through assisted traceability," *IEEE Trans. Softw. Eng.*, vol. 49, no. 12, pp. 5060–5083, Oct. 2023.

[16] L. Fan *et al.*, "A bibliometric review of large language models research from 2017 to 2023," *ArXiv*, 2023.

[17] W. Chen, "Large language models are few(1)-shot table reasoners," *ArXiv*, 2022.

[18] W. X. Zhao *et al.*, "A survey of large language models," *ArXiv*, 2023.

[19] G. Yenduri *et al.*, "Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions," *ArXiv*, vol. abs/2305.10435, 2023.

[20] C. A. G. Cano, V. S. Castillo, and T. A. C. Gallego, "Unveiling the thematic landscape of generative pre-trained transformer (gpt) through bibliometric analysis," *Metaverse Basic and Applied Research*, pp. 1–8, 2023.

[21] A. S. Gill, "Chat generative pretrained transformer: Extinction of the designer or rise of an augmented designer," in *Int. Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2023, pp. 1–5.

[22] Q. Zhu and J. Luo, "Generative pre-trained transformer for design concept generation: An exploration," *Proceedings of the Design Society*, vol. 2, pp. 1825–1834, 2021.

[23] Y.-G. Guéhéneuc, "P-mart: Pattern-like micro architecture repository," *Proceedings of the 1st EuroPLoP Focus Group on pattern repositories*, pp. 1–3, 2007.

Patterns for Quantum Machine Learning

Lavinia Stiliadou, Johanna Barzen, Martin Beisel, Frank Leymann, and Benjamin Weder
Institute of Architecture of Application Systems, *University of Stuttgart*
Universitätsstrasse 38, 70569 Stuttgart, Germany
e-mail: {lastname}@iaas.uni-stuttgart.de

***Abstract*—The tremendous success of applying machine learning techniques in science and industry leads to new challenges: Developing new, faster, or more precise algorithms can not compete with the continuously growing data volumes to process. Thus, the computational resources must be increased, leading to high costs and energy consumption. To overcome these issues, quantum machine learning promises to utilize quantum mechanical phenomena to train machine learning models more efficiently. However, realizing such quantum machine learning techniques is time-consuming and complex. This is especially the case as new techniques are typically published as scientific papers without suitable documentation for software developers. Patterns are a well-established concept to document proven solutions to recurring problems. Although a pattern language for quantum computing has been introduced, it currently misses patterns documenting how quantum machine learning can be successfully applied. To bridge this gap, this paper presents three novel patterns, focusing on quantum machine learning techniques.**

***Keywords*-Quantum Computing; Pattern Language; QML.**

I. Introduction

In recent years, machine learning has revolutionized the industry by providing new means for solving problems in various domains, e.g., natural language processing or medicine [1]. However, a significant part of this progress was achieved by increasing the computational resources [2]. Thus, costs and energy consumption for reaching new milestones in the machine learning domain increased dramatically. For example, the training of modern neural networks, such as GPT4 [3], takes months and costs millions of dollars [4]. Quantum machine learning provides new concepts and algorithms taking advantage of quantum mechanical phenomena, such as superposition and entanglement, that promise to more efficiently train machine learning models [5][6]. Hence, quantum machine learning vastly differs from classical machine learning, e.g., in the structuring of the training data [7][8]. However, these concepts of quantum machine learning are complex, particularly for software developers without deep knowledge of physics and quantum computing. Additionally, there currently exists no structured documentation on how software developers can employ quantum machine learning to benefit from its advantages.

A well-established concept for documenting proven solutions to recurring problems in the software engineering domain is patterns [9][10]. Patterns facilitate understanding the origin of a problem and the forces that complicate solving it. Furthermore, they describe an abstract solution to the problem, enabling software developers to implement and customize the presented solution strategy for their given application domain.

To support developers in building quantum applications, Leymann [11] has introduced the quantum computing pattern language, which was continuously expanded to cover different areas of quantum computing. It documents fundamental concepts of quantum computing and discusses various techniques, e.g., error handling [12] or warm-starting [13]. While there are already patterns for solving optimization problems [14], there are no patterns documenting how quantum machine learning techniques can be successfully applied. In this paper, we extend the quantum computing pattern language by documenting three novel patterns for quantum machine learning, describing well-known types of quantum algorithms.

The remainder of this paper is structured as follows: In Section II, we introduce fundamentals about quantum algorithms and hybrid quantum applications. Furthermore, we present the utilized pattern format, as well as the authoring method. Section III extends the quantum computing pattern language by documenting the newly introduced quantum machine learning patterns. In Section IV, the presented quantum machine learning patterns are discussed and evaluated. Finally, Section V presents related work, and Section VI concludes the paper.

II. Fundamentals

In this section, we discuss the fundamentals of quantum algorithms and their use in quantum applications. Furthermore, we present our pattern format and the pattern authoring method.

A. Quantum Algorithms & Applications

Quantum computing achieves speed-ups compared to classical computations by leveraging quantum mechanical phenomena [15]: Unlike classical computers that operate on 0 s and 1 s, quantum devices utilize qubits that can exist in a superposition of states $|0\rangle := (1,0)^T$ and $|1\rangle := (0,1)^T$. Further, qubits can be entangled with each other, making their states inseparable. Various technologies to realize quantum devices exist, e.g., superconducting or photonic quantum devices, that utilize different computation models for quantum computing. In this paper, we focus on the gate-based computation model, which is used by quantum hardware providers such as IBM and Rigetti. To perform computations on gate-based quantum devices, the states of the qubits are manipulated utilizing different quantum gates, which are specified by a quantum program, a so-called quantum circuit [16]. There are single-qubit gates, which only affect the state of a single qubit, and multi-qubit gates, which affect multiple qubits and can entangle multiple qubits with each other. To retrieve a classical solution when executing

a quantum circuit, the state of the quantum system must be measured. Since the state of the system collapses when measuring it and quantum computing is probabilistic by nature, quantum executions are typically executed multiple times to retrieve the expectation value of the computation [17].

Quantum algorithms are often hybrid, comprising classical pre- and post-processing steps, e.g., encoding data into a quantum circuit or mitigating occurring errors [16][18]. A special class of quantum algorithms is so-called *Variational Quantum Algorithms (VQAs)* [19]. VQAs typically utilize shallow circuits, i.e., they have a low number of consecutive gates and use a small number of qubits. Therefore, they enable the execution of practically relevant quantum computations on today's error-prone and intermediate-size quantum devices. The execution of VQAs alternates between executing parameterized quantum circuits and classically optimizing their parameters, e.g., using gradient-based or gradient-free optimization approaches. In each iteration of the VQA, the execution results are used to estimate the value of the cost function that encodes the problem. Its value is utilized by an optimizer to find more suitable circuit parameters, e.g., by minimizing the cost function. The quantum cost function is generally defined as: $C(\vec{\theta}) = f(\{\rho_i\}_{i=1}^{k}, \{O_i\}_{i=1}^{k}, U(\vec{\theta}))$, where $\vec{\theta}$ is a vector of parameters, f is some function, k is the size of the training set T. The set $\{\rho_i\}$ represents the input states from the training set, while $\{O_i\}$ denotes the set of observables. The parametrized quantum circuit $U(\vec{\theta})$, defined by the parameters $\vec{\theta}$, is iteratively optimized classically.

B. Patterns & Authoring Method

Patterns are an established concept for documenting proven solutions to commonly recurring problems in a well-structured manner [9]. Typically, a uniform pattern format is used for all patterns of a domain to facilitate understanding the patterns within a pattern language [20]. Therefore, we use the pattern format utilized by the previously published quantum computing patterns, which is structured as follows: Each pattern is identified by a unique *name* within the pattern language. Furthermore, patterns are associated with a mnemonic *icon* to enable a visual recognition of the pattern. The problem solved by the pattern is concisely summarized in a *problem* statement. Next, the *context* in which the problem appears is described and the *forces* complicating the solution of the problem are discussed. In the *solution* section, a proven strategy for solving the previously discussed problem is presented alongside a corresponding *solution sketch*. Subsequently, *examples* of the solution are explained. The *result* section describes the consequences of applying the solution and discusses what further steps might be necessary to handle them. Each pattern is semantically linked to *related patterns* in the eponymous section, e.g., to patterns that are commonly used in combination or that are alternatives to each other. Finally, the *known uses* section showcases real-world occurrences of the pattern. To identify and document the patterns for quantum machine learning, we applied the pattern authoring method introduced by Fehling et al. [20]. First, we analyzed the literature, as well as the documentation of current quantum software development kits for best practices and established solution strategies for quantum machine learning. Subsequently, the collected information was filtered based on its relevancy for practically applying quantum machine learning. The identified solutions were documented and iteratively refined to extract patterns comprising quantum machine learning algorithms. For the documentation of patterns, their previously described format is utilized.

III. Quantum Machine Learning Patterns

In this section, we first provide a brief overview of the existing quantum computing pattern language and subsequently introduce three novel patterns for quantum machine learning.

A. Quantum Computing Pattern Language Overview

Figure 1 gives an overview of the quantum computing pattern language. It comprises both the already existing, as well as the newly added patterns from this work. The different patterns are assigned to one category, depending on the phase of the hybrid quantum application lifecycle they belong to [18]:

First, the *unitary transformations* patterns [21] describe best practices for transformations after an initial state has been created. The *warm-starting* patterns [13] show various techniques to improve the performance of quantum algorithms. Next, the *program flow* patterns [14] summarize concepts to split computations between quantum and classical hardware. The *circuit cutting* patterns [22] document techniques to cut large quantum circuits into smaller circuits that can be successfully executed on today's quantum devices. To encode classical data into quantum circuits, the *data encodings* patterns [21] describe so-called state preparation routines. The *error handling* patterns [12] summarize approaches to reduce noise on today's quantum devices. Fundamental *quantum states* [11], how they are created, and for which quantum algorithms they are used as a basis are discussed in the eponymous category. For executing quantum circuits and hybrid quantum applications, different quantum cloud offerings are available. These offerings provide heterogeneous features, e.g., the execution via a queue or the exclusive reservation of a quantum device, and the *execution* patterns [23] document these execution styles, as well as their benefits and disadvantages. The *development* patterns [24] provide solutions and best practices for typical problems when developing hybrid quantum applications. Complementary, the *operations* patterns [25] cover abstract solutions for operating and managing hybrid quantum applications. Finally, the *measurement* patterns [21] present concepts and techniques for extracting classical data from quantum states.

In this work, we introduce three novel patterns describing well-known approaches for tackling crucial problems from the quantum machine learning domain: To overcome the difficulties when clustering large and complex data sets, Quantum Clustering algorithms have been introduced. They partition data sets into different clusters based on their similarity, which is calculated using quantum devices. The Quantum Classification pattern provides a means for

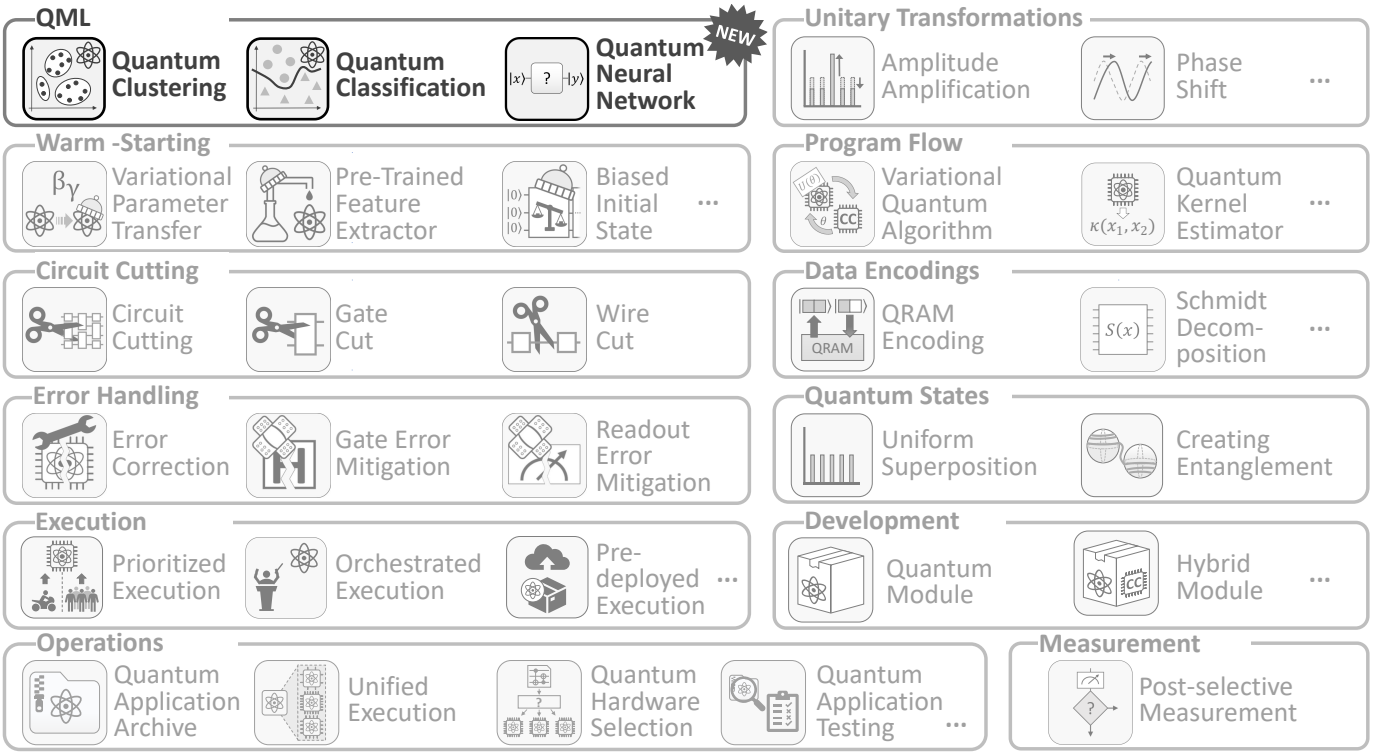
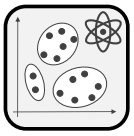


Figure 1. Overview of the quantum computing pattern language with some existing (light gray) and the new patterns proposed in this work (dark gray).

training a classifier based on a set of labeled data. This classifier enables assigning new data points to one of multiple classes. Finally, the QUANTUM NEURAL NETWORK pattern documents how quantum devices can be used to improve the accuracy and performance of neural networks by leveraging quantum mechanical phenomena.

B. Quantum Clustering



Problem: How to partition a data set into different clusters based on their similarity utilizing a quantum device?

Context: A set of unlabeled data needs to be grouped into different clusters. The clusters should organize the data points according to identified similarities.

Forces: Data sets may exhibit non-linear separability, which increases the complexity when clustering. Moreover, data sets utilized in machine learning continuously grow in size, leading to increased training times [3]. Therefore, algorithms whose runtime scales well with the number of data points in the data set and the dimensions of the feature space are required. In addition to the general machine learning forces, also quantum-specific forces have to be taken into account. For example, loading large data sets consisting of many tuples into current quantum devices is difficult due to the high circuit depth of the required state preparation routines [26][27].

Solution: Figure 2 gives an overview of the general clustering process. Use a quantum device to cluster the m data points $\{x_i\}_{i=1}^{m}$ of a data set. First, the classical data points are encoded into quantum states $\{|x_i\rangle\}_{i=1}^{m}$ by applying a unitary transformation U_ϕ, enabling the quantum computer to process the data. Once the data points are encoded, a given ansatz $V(\vec{\theta})$ is used to calculate the similarity between data points. The ansatz is a parameterized quantum circuit designed to approximate the quantum state that captures the relevant features of the data for similarity measurement. The ansatz computes the similarity either between pairwise data points or between all data points, depending on its structure [28]. The cost function used in this approach is designed to assign similar points to the same cluster and points with low similarity to different clusters. In the cost function, this is represented by a penalty term that penalizes distant points that are assigned to the same cluster. Additionally, the clustering process is controlled by adding constraints. To ensure that each data point is assigned to exactly one cluster, the following condition must hold $\sum_{a=1}^{k} q_i^a = 1$. Thereby, classical variables q_i^a are introduced to denote whether a data point x_i is assigned to cluster a. The quantum circuit parameters are updated iteratively to minimize the cost function.

Examples: An exemplary quantum clustering algorithm is the quantum k-means algorithm [29][30]: First, k initial data points are randomly selected as centroids for the clustering. Then, the states for both the centroids, as well as the remaining data points are prepared. The number k, which corresponds to the number of clusters, can either be specified by the user or automatically determined [31]. Subsequently, for each data point, the distance to all centroids is calculated utilizing a distance metric, e.g.,

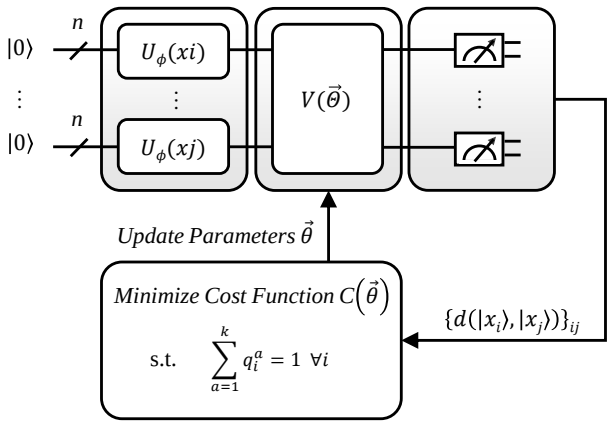


Figure 2. Solution sketch for the QUANTUM CLUSTERING pattern.

the Manhattan distance [30] or Euclidean distance [32]. For example, the SWAP test [33] can be used to determine the Euclidean distances efficiently on a quantum device. Each data point is assigned to the cluster corresponding to the centroid with the smallest distance to the data point. Afterward, the new centroids are calculated classically by computing the mean of all data points assigned to that cluster. If the retrieved centroids differ substantially, i.e., more than a certain threshold specified by the user, from the previous iteration, the previously described procedure is performed again utilizing the new centroids.

Result: Utilizing a quantum clustering algorithm may enable identifying clusters in a data set exponentially faster than with a classical clustering algorithm [32]. Often, the computational advantage of quantum clustering algorithms relies on the availability of the input data in a suitable format. Once an implementation of *Quantum Random Access Memory (QRAM)* is available, data can be encoded efficiently, enabling the full potential of quantum clustering.

Related Patterns: The QRAM ENCODING pattern [21] can be used to efficiently encode the data points for a quantum device. To facilitate the clustering of complex data sets, the data points can be mapped into a higher dimensional feature space using the QUANTUM KERNEL ESTIMATOR pattern. The QUANTUM CLUSTERING pattern uses the QUANTUM-CLASSICAL SPLIT pattern [11] to efficiently distribute the computations using quantum and classical hardware and can be realized as a HYBRID MODULE [24].

Known Uses: Ramirez [34] presents different quantum clustering techniques, such as quantum spectral clustering and quantum hierarchical clustering. Kavitha et al. [35] utilize quantum k-means clustering for detecting heart diseases. Patil et al. [36] introduce two measurement-based quantum clustering algorithms. The first algorithm follows a hierarchical clustering approach. The second algorithm uses unsharp measurements for the clustering process. Gopalakrishnan et al. [37] propose a quantum clustering algorithm that achieves linear scalability with respect to both the number of data points and their density.

C. Quantum Classification

Problem: How to train a classifier to assign new data points to one of multiple classes using a quantum device?

Context: New data points need to be classified into one of several different classes. A labeled set of training data is given.

Forces: Classifying data is getting increasingly more difficult when the feature space becomes larger [38]. While quantum computing enables solving this problem by utilizing efficient quantum algorithms, it also leads to additional challenges. For example, high-dimensional data sets can lead to large quantum circuits that may not be successfully executable on today's *Noisy Intermediate-Scale Quantum (NISQ)* devices [27]. Additionally, quantum approaches can suffer from exponential cost concentration, which makes models less sensitive to input data, leading to generalization problems [39][40].

Solution: Train a classifier using a quantum device to classify new data points precisely. In Figure 3, an overview of two different approaches for training a classifier is depicted. Classifiers can either be trained using (i) a kernel-based method or (ii) a variational method. Generally, the input for training a classifier is an initial set of labeled data $\{(x_i, y_i)\}_{i=1}^{n}$, where x_i are the feature vectors, y_i are the labels, i.e., real numbers, and n is the size of the training set. In the kernel-based approach, a quantum kernel is used to measure the similarities between data points by mapping them into a high-dimensional Hilbert space and computing the inner product of their corresponding quantum states. This quantum kernel is computed for all pairs of training data by applying a unitary $U_\phi(x_i)$ to encode each data point x_i into a quantum state. The adjoint operation $U_\phi^\dagger(x_j)$ is then applied to calculate the overlap between the states corresponding to x_i and x_j. Then, a classical algorithm, e.g., a classical support vector machine [41], is used for computing the classifier based on the previously calculated kernel. Alternatively, the variational method optimizes the parameters of a quantum circuit to directly realize the classifier. In this approach, a data point x is first encoded into a quantum state using a unitary $U_\phi(x)$, which maps the classical data into

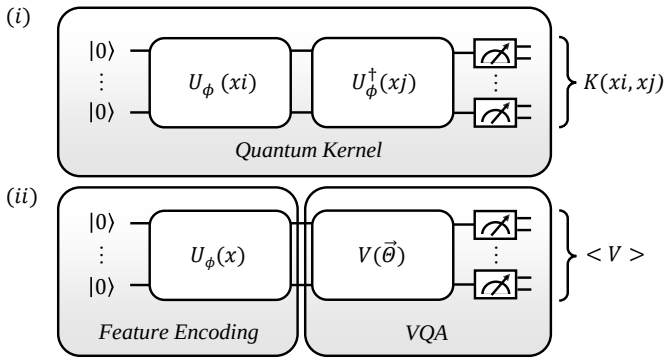


Figure 3. Solution sketch for the QUANTUM CLASSIFICATION pattern.

a quantum state. Once the data are encoded, a parameterized quantum circuit is applied. The circuit produces an output whose expectation value $< V >$ determines the predicted label for a given data point x. The parameters of the circuit are iteratively optimized by a classical optimizer that minimizes a quantum cost function that calculates the differences between the predicted and actual labels from the data set.

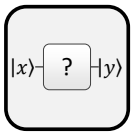
Examples: An example is the kernel-based quantum support vector machine [42], achieving logarithmic complexity with respect to both the data dimension and the number of training examples. Another example is the variational quantum support vector machine [38][43], which uses a parameterized quantum circuit to directly implement a SVM on a QPU.

Result: After the training process, the classifier can be used to assign new data points to one of the existing classes. Utilizing a quantum classifier may enable training a more accurate classifier than using a classical classification technique [44]. Quantum classifiers still function under the influence of noise as they are resistant to a small number of misclassifications [38]. This is particularly important with the noisiness of today's quantum devices. However, mitigation mechanisms must be implemented to address the challenges, such as cost concentration in kernel values or flatness in the optimization landscape [19][39].

Related Patterns: The QRAM ENCODING pattern [21] can be utilized to achieve a speed-up when encoding data. A quantum classifier can be realized using a VARIATIONAL QUANTUM ALGORITHM [14].

Known Uses: To train the quantum classifier, different approaches can be used, e.g., variational quantum support vector machines [34][38], quantum decision trees [45], and quantum nearest neighbor classification [46]. Furthermore, quantum classifiers have been applied in different application areas, e.g., image recognition [47], analyzing the sentiments of sentences [48], and predicting air pollution [49].

D. Quantum Neural Network (QNN)



Problem: How to learn an unknown unitary operator using a quantum device?

Context: An unknown unitary operator needs to be learned from a training set containing the quantum inputs and the expected quantum outputs.

Forces: Identifying a unitary operator that is capable of mapping input data to their respective output is getting increasingly more difficult with the complexity and variety of the data. Determining such a mapping requires a lot of input data, and the training procedure requires significant computational power [3]. However, this number can be reduced as outlined by the Quantum No-Free-Lunch Theorem since only obtaining a subset of the training samples as entangled quantum states is already beneficial [50].

Solution: Figure 4 shows the training process of a QNN to learn an unknown unitary operator U: To realize a corresponding quantum circuit, first, the input data are encoded to a quantum state $|x\rangle$. Similarly to classical neural networks, quantum circuits realizing a QNN comprise various parameterized hidden layers $V_i(\vec{\theta})$ to approximate U. The parameters $\vec{\theta}$ are iteratively adjusted by an optimizer, which minimizes a cost function until the quantum circuit produces approximately the expected outputs. The cost function uses the expected outputs and similarity measures, such as fidelity, to evaluate how closely the produced outputs $|\tilde{y}\rangle$ match the expected ones $|y\rangle$.

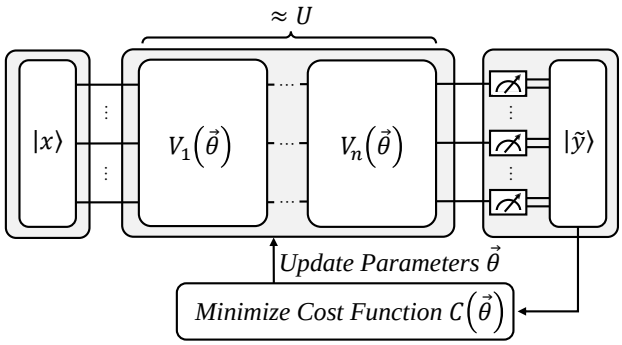


Figure 4. Solution sketch for the QUANTUM NEURAL NETWORK pattern.

Examples: A special kind of QNNs are quantum convolutional neural networks, which are utilized for processing structured grid data, e.g., for image processing [51][52]. It comprises four different types of layers: (i) First, state preparation layers are used to encode the classical input data. (ii) The convolutional layers enable the detection of spatial patterns within the input data. (iii) Pooling layers reduce some of the spatial dimensions to focus on the optimization of the most important features. (iv) Finally, fully connected layers are used to produce the final output of the quantum convolutional network.

Result: The result is a set of parameters that configures the QNN to approximate the expected output of the unknown unitary operator. The quality of the approximation depends on the size of the training set, its linear structure, and the degree of entanglement [53]. While entangled data provides benefits for the training of QNNs, a too high level of entanglement can lead to barren plateaus [54].

Related Patterns: Quantum neural networks are a realization of the VARIATIONAL QUANTUM ALGORITHM pattern [14]. Different state preparation routines, such as ANGLE ENCODING or BASIS ENCODING [21], can be utilized to encode the input data of the QNN. To integrate a QNN in existing applications, it can be provided as a HYBRID MODULE [24].

Known Uses: Jeswal et al. [55] and Vasuki et al. [56] provide surveys overviewing the various application areas of QNNs, ranging from prediction to pattern recognition problems. Kashif et al. [57] present an approach for efficiently training QNNs in the presence of noise when using NISQ devices.

IV. DISCUSSION

In this section, we discuss the challenges and limitations of applying the presented quantum machine learning patterns and elaborate on the validity of the documented patterns.

Quantum machine learning is a rapidly evolving field that promises to overcome the limitations of state-of-the-art

classical machine learning methods [58]. However, the high error rates and low number of qubits of today's quantum devices prevent the application of the introduced concepts for many real-world problems. The roadmaps of quantum device providers, such as IBM [59] and QuEra [60], promise that in the near future, the potential of quantum machine learning can be demonstrated for practically relevant use cases.

An essential part of achieving speed-ups with quantum devices is efficient access to classical and quantum data [54]. Thereby, the classical data is prepared for the quantum device utilizing state preparation routines [21]. While there are efficient implementations for many state preparation routines, QRAM has not yet been successfully implemented [61]. This limits the effectiveness of many quantum machine learning algorithms, such as the quantum support vector machine, as the assumption of the algorithms is that data is available via QRAM [38]. Therefore, alternative, less efficient state preparation routines must be utilized until an efficient QRAM implementation is available.

To confirm the validity of patterns in the software engineering domain, a number of different real-world implementations of the patterns are identified [62]. Hence, several occurrences of each quantum machine learning pattern have been documented in the known uses section of the respective pattern. To ease the configuration and abstract technical details, patterns can be used to automate the generation of quantum applications [63].

V. Related Work

The patterns for quantum machine learning introduced in this work extend the existing quantum computing pattern language presented in Section III-A. Perez-Castillo et al. [64] analyze code repositories for the occurrences of different patterns of the quantum computing pattern language and identify a lack of abstraction mechanisms. The patterns presented in this paper aim to bridge this gap in the quantum machine learning domain. Aside from the quantum computing patterns, there are other works presenting patterns in the quantum computing domain that do not follow the pattern format introduced by Alexander et al. [9]: Baczyk et al. [65] document different patterns that aim to facilitate architectural design decisions when building quantum applications. Khan et al. [66] identify various architecture design patterns for quantum applications via a systematic literature survey. Huang and Martonosi [67] utilize quantum programming patterns to find bugs in quantum circuits. Gilliam et al. [68] and Perdrix [69] present patterns for building quantum circuits. However, none of these papers focus on patterns in the quantum machine learning domain.

Guo et al [70] present a set of patterns for defining ansätze in variational quantum algorithms. These ansätze can also be used in quantum machine learning, e.g., to implement the hidden layers of quantum neural networks.

Lakshmanan et al. [71] document various machine learning design patterns. They focus on different aspects that should be regarded when utilizing machine learning in practice, e.g., reproducibility or responsible artificial intelligence. Although the machine learning design patterns do not consider quantum machine learning, various patterns, such as Model Versioning can also be applied to quantum machine learning.

Falkenthal et al. [72] introduce the concept of solution languages to facilitate the application of patterns for real-world use cases. Solution languages comprise so-called concrete solutions, i.e., implementations of a pattern for a specific use case, e.g., a quantum circuit or a Python program. These concrete solutions are associated with the corresponding pattern, enabling developers to reuse existing implementations for their applications. Thereby, the manual effort of implementing the abstract solution described by the pattern can be reduced.

VI. Conclusion & Future Work

Machine learning has revolutionized research and industry by providing new means for solving various problems. However, a significant part of this progress was achieved by increasing the computational resources, leading to high costs and energy consumption. A promising technology providing additional computational power is quantum computing. In this paper, we capture existing concepts from the literature for utilizing quantum devices to tackle crucial machine learning problems efficiently. We document these concepts in an easily digestible manner as patterns that enable understanding the problem and solving it using proven solution strategies. The introduced patterns are publicly available via Pattern Atlas [73], an open-source tool for authoring, managing, and visualizing patterns [74].

Since quantum machine learning is a rapidly evolving and highly active area, we will continue investigating the progress achieved by researchers and companies. Hence, in future work, we plan to identify new solution strategies in the quantum machine learning domain and document them as novel patterns.

ACKNOWLEDGEMENTS

This work was partially funded by the BMWK projects *EniQmA* (01MQ22007B) and *SeQuenC* (01MQ22009B).

References

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects", *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The Computational Limits of Deep Learning", *arXiv:2007.05558*, 2020.

[3] OpenAI, "GPT-4 Technical Report", *arXiv:2303.08774*, 2023.

[4] B. Cottier, R. Rahman, L. Fattorini, N. Maslej, and D. Owen, "The rising costs of training frontier AI models", *arXiv:2405.21015*, 2024.

[5] J. Biamonte *et al.*, "Quantum Machine Learning", *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[6] J. Barzen, "From Digital Humanities to Quantum Humanities: Potentials and Applications", in *Quantum Computing in the Arts and Humanities: An Introduction to Core Concepts, Theory and Applications*, Springer, 2022, pp. 1–52.

[7] T. Gabor *et al.*, "The Holy Grail of Quantum Artificial Intelligence: Major Challenges in Accelerating the Machine Learning Pipeline", in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 456–461.

[8] S. Thanasilp, S. Wang, N. A. Nghiem, P. Coles, and M. Cerezo, "Subtleties in the trainability of quantum machine learning models", *Quantum Machine Intelligence*, vol. 5, no. 1, p. 21, 2023.

[9] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[10] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, 2004.

[11] F. Leymann, "Towards a Pattern Language for Quantum Algorithms", in *Proceedings of the 1st International Workshop on Quantum Technology and Optimization Problems (QTOP)*, Springer, 2019.

[12] M. Beisel *et al.*, "Patterns for Quantum Error Handling", in *Proceedings of the 14th International Conference on Pervasive Patterns and Applications (PATTERNS)*, Xpert Publishing Services (XPS), 2022, pp. 22–30.

[13] F. Truger, J. Barzen, M. Beisel, F. Leymann, and V. Yussupov, "Warm-Starting Patterns for Quantum Algorithms", in *Proceedings of the 16th International Conference on Pervasive Patterns and Applications (PATTERNS)*, Xpert Publishing Services (XPS), 2024, pp. 25–31.

[14] M. Weigold, J. Barzen, F. Leymann, and D. Vietz, "Patterns for Hybrid Quantum Algorithms", in *Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC)*, Springer, 2021, pp. 34–51.

[15] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[16] 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, pp. 1–28, 2020.

[17] F. Leymann *et al.*, "Quantum in the Cloud: Application Potentials and Research Opportunities", in *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER)*, SciTePress, 2020, pp. 9–24.

[18] B. Weder, J. Barzen, F. Leymann, and D. Vietz, "Quantum Software Development Lifecycle", in *Quantum Software Engineering*, Springer, 2022, pp. 61–83.

[19] M. Cerezo *et al.*, "Variational quantum algorithms", *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[20] C. Fehling, J. Barzen, U. Breitenbücher, and F. Leymann, "A Process for Pattern Identification, Authoring, and Application", in *Proceedings of the 19th European Conference on Pattern Languages of Programs (EuroPLoP)*, ACM, 2014.

[21] 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.

[22] M. Bechtold, J. Barzen, M. Beisel, F. Leymann, and B. Weder, "Patterns for Quantum Circuit Cutting", in *Proceedings of the 30th Conference on Pattern Languages of Programs (PLoP)*, HILLSIDE, 2023.

[23] D. Georg *et al.*, "Execution Patterns for Quantum Applications", in *Proceedings of the 18th International Conference on Software Technologies (ICSOFT)*, SciTePress, 2023, pp. 258–268.

[24] F. Bühler *et al.*, "Patterns for Quantum Software Development", in *Proceedings of the 15th International Conference on Pervasive Patterns and Applications (PATTERNS)*, Xpert Publishing Services (XPS), 2023, pp. 30–39.

[25] M. Beisel, J. Barzen, F. Leymann, and B. Weder, "Operations Patterns for Hybrid Quantum Applications", in *Proceedings of the 15th International Conference on Cloud Computing and Services Science (CLOSER)*, SciTePress, 2025.

[26] V. Akshay, H. Philathong, M. Morales, and J. Biamonte, "Reachability deficits in quantum approximate optimization", *Physical review letters*, vol. 124, no. 9, p. 090 504, 2020.

[27] J. Preskill, "Quantum Computing in the NISQ era and beyond", *Quantum*, vol. 2, p. 79, 2018.

[28] A. Poggiali, A. Berti, A. Bernasconi, G. M. Del Corso, and R. Guidotti, "Quantum clustering with k-means: A hybrid approach", *Theoretical Computer Science*, vol. 992, p. 114 466, 2024.

[29] S. U. Khan, A. J. Awan, and G. Vall-Llosera, "K-Means Clustering on Noisy Intermediate Scale Quantum Computers", *arXiv:1909.12183*, 2019.

[30] Z. Wu, T. Song, and Y. Zhang, "Quantum k-means algorithm based on Manhattan distance", *Quantum Information Processing*, vol. 21, no. 1, p. 19, 2021.

[31] I. Khan, Z. Luo, J. Z. Huang, and W. Shahzad, "Variable Weighting in Fuzzy k-Means Clustering to Determine the Number of Clusters", *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 9, pp. 1838–1853, 2019.

[32] S. U. Khan, A. J. Awan, and G. Vall-Llosera, "K-Means Clustering on Noisy Intermediate Scale Quantum Computers", *arXiv:1909.12183*, 2019.

[33] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf, "Quantum Fingerprinting", *Physical review letters*, vol. 87, no. 16, p. 167 902, 2001.

[34] J. G. C. Ramírez, "Advanced Quantum Algorithms for Big Data Clustering and High-Dimensional Classification", *Journal of Advanced Computing Systems*, vol. 4, no. 6, 2024.

[35] S. S. Kavitha and N. Kaulgud, "Quantum K-means clustering method for detecting heart disease using quantum circuit approach", *Soft Computing*, vol. 27, no. 18, pp. 13 255–13 268, 2023.

[36] S. Patil, S. Banerjee, and P. K. Panigrahi, "Measurement-Based Quantum Clustering Algorithms", *arXiv:2302.00566*, 2023.

[37] D. Gopalakrishnan *et al.*, "qCLUE: a quantum clustering algorithm for multi-dimensional datasets", *Frontiers in Quantum Science and Technology*, 2024.

[38] V. Havlíček *et al.*, "Supervised learning with quantum-enhanced feature spaces", *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[39] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes, "Exponential concentration in quantum kernel methods", *Nature Communications*, vol. 15, no. 1, p. 5200, 2024.

[40] A. Arrasmith, Z. Holmes, M. Cerezo, and P. J. Coles, "Equivalence of quantum barren plateaus to cost concentration and narrow gorges", *Quantum Science and Technology*, vol. 7, no. 4, p. 045 015, 2022.

[41] C. J. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[42] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum Support Vector Machine for Big Data Classification", *Physical review letters*, vol. 113, no. 13, p. 130 503, 2014.

[43] G. Gentinetta, A. Thomsen, D. Sutter, and S. Woerner, "The complexity of quantum support vector machines", *Quantum*, vol. 8, p. 1225, 2024.

[44] S. S. Kavitha and N. Kaulgud, "Quantum machine learning for support vector machine classification", *Evolutionary Intelligence*, vol. 17, no. 2, pp. 819–828, 2024.

[45] S. Lu and S. L. Braunstein, "Quantum decision tree classifier", *Quantum information processing*, vol. 13, no. 3, pp. 757–770, 2014.

[46] W. Roga, B. Chevalier, and M. Takeoka, "Fully Quantum Classifier", *arXiv:2307.03396*, 2023.

[47] J. Li, Y. Li, J. Song, J. Zhang, and S. Zhang, "Quantum Support Vector Machine for Classifying Noisy Data", *IEEE Transactions on Computers*, vol. 73, no. 9, pp. 2233–2247, 2024.

[48] F. Z. Ruskanda, M. R. Abiwardani, R. Mulyawan, I. Syafalni, and H. T. Larasati, "Quantum-Enhanced Support Vector Ma-

chine for Sentiment Classification", *IEEE Access*, vol. 11, pp. 87 520–87 532, 2023.
[49] O. Farooq *et al.*, "An enhanced approach for predicting air pollution using quantum support vector machine", *Scientific Reports*, vol. 14, no. 1, p. 19 521, 2024.
[50] A. Mandl, J. Barzen, F. Leymann, and D. Vietz, "On Reducing the Amount of Samples Required for Training of QNNs: Constraints on the Linear Structure of the Training Data", *arXiv:2309.13711*, 2023.
[51] T. Hur, L. Kim, and D. K. Park, "Quantum convolutional neural network for classical data classification", *Quantum Machine Intelligence*, vol. 4, no. 1, p. 3, 2022.
[52] S. Wei, Y. Chen, Z. Zhou, and G. Long, "A quantum convolutional neural network on NISQ devices", *AAPPS Bulletin*, vol. 32, pp. 1–11, 2022.
[53] A. Mandl *et al.*, "Linear Structure of Training Samples in Quantum Neural Network Applications", in *International Conference on Service-Oriented Computing*, Springer, 2023, pp. 150–161.
[54] S. Thanasilp, S. Wang, N. A. Nghiem, P. Coles, and M. Cerezo, "Subtleties in the trainability of quantum machine learning models", *Quantum Machine Intelligence*, vol. 5, no. 1, 2023.
[55] S. K. Jeswal and S. Chakraverty, "Recent Developments and Applications in Quantum Neural Network: A Review", *Archives of Computational Methods in Engineering*, vol. 26, no. 4, pp. 793–807, 2019.
[56] M. Vasuki, A. Karunamurthy, R. Ramakrishnan, and G. Prathiba, "Overview of Quantum Computing in Quantum Neural Network and Artificial Intelligence", *Quing International Journal of Innovative Research in Science and Engineering*, vol. 2, pp. 117–127, 2023.
[57] M. Kashif and M. Shafique, "HQNET: Harnessing Quantum Noise for Effective Training of Quantum Neural Networks in NISQ Era", *arXiv:2402.08475*, 2024.
[58] K. A. Tychola, T. Kalampokas, and G. A. Papakostas, "Quantum Machine Learning—An Overview", *Electronics*, vol. 12, no. 11, p. 2379, 2023.
[59] IBM, *Quantum Roadmap*, https://www.ibm.com/roadmaps/quantum, 2024, [accessed: 2025.03.01].
[60] QuEra, *Our Quantum Computing Roadmap*, https://www.quera.com/qec, 2024, [accessed: 2025.03.01].
[61] S. Jaques and A. G. Rattew, "QRAM: A Survey and Critique", *arXiv:2305.10310*, 2023.
[62] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley, 1995.
[63] D. Vietz *et al.*, "Harnessing Patterns to Support the Development of Hybrid Quantum Applications", in *Proceedings of the 14th International Conference on Software and Computer Applications (ICSCA)*, ACM, 2025, pp. 1–11.
[64] R. Pérez-Castillo, M. Fernández-Osuna, J. A. Cruz-Lemus, and M. Piattini, "A preliminary study of the usage of design patterns in quantum software", in *Proceedings of the 5th ACM/IEEE International Workshop on Quantum Software Engineering (QSE)*, 2024, pp. 41–48.
[65] M. Baczyk, R. Pérez-Castillo, and M. Piattini, "Towards a Framework of Architectural Patterns for Quantum Software Engineering", in *Proceedings of the 4th International Workshop on Quantum Software Engineering and Technology (Q-SET)*, vol. 2, 2024, pp. 228–233.
[66] A. A. Khan *et al.*, "Software architecture for quantum computing systems — A systematic review", *Journal of Systems and Software*, vol. 201, p. 111 682, 2023.
[67] 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 (ISCA)*, ACM, 2019, pp. 541–553.
[68] A. Gilliam *et al.*, "Foundational Patterns for Efficient Quantum Computing", *arXiv:1907.11513*, 2019.
[69] S. Perdrix, "Quantum Patterns and Types for Entanglement and Separability", *Electronic Notes in Theoretical Computer Science*, vol. 170, pp. 125–138, 2007.
[70] X. Guo, T. Muta, and J. Zhao, "Quantum Circuit Ansatz: Patterns of Abstraction and Reuse of Quantum Algorithm Design", *arXiv:2405.05021*, 2024.
[71] V. Lakshmanan, S. Robinson, and M. Munn, *Machine learning design patterns*. O'Reilly Media, 2020.
[72] M. Falkenthal, J. Barzen, U. Breitenbücher, and F. Leymann, "Solution Languages: Easing Pattern Composition in Different Domains", *International Journal on Advances in Software*, vol. 10, no. 3, pp. 263–274, 2017.
[73] Kipu Quantum, *PlanQK Pattern Atlas*, https://patterns.platform.planqk.de/pattern-languages/af7780d5-1f97-4536-8da7-4194b093ab1d, 2025, [accessed: 2025.03.01].
[74] F. Leymann and J. Barzen, "Pattern Atlas", in *Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future*, Springer, 2021, pp. 67–76.

Using the Monte Carlo Method to Estimate Student Motivation in Scientific Computing

Isaac Caicedo-Castro*†‡ Oswaldo Vélez-Langs*‡ Rubby Castro-Púche†§
*Socrates Research Team
†Research Team: Development, Education, and Healthcare
‡Faculty of Engineering
§Faculty of Education and Human Science
University of Córdoba
Carrera 6 No. 76-103, 230002, Montería, Colombia
e-mail: {isacaic | oswaldovelez | rubycastro}@correo.unicordoba.edu.co

***Abstract*—In this study, we investigate students' motivation to learn scientific computing in the undergraduate systems engineering program at the University of Córdoba (Colombia). Scientific computing is often a challenging subject for college students; therefore, motivation plays a crucial role in succeeding in these courses. To quantify the factors that potentially impact student motivation, we conducted a survey of 117 students, relying on their perceptions. Using an F-test, we selected 15 independent variables from the original set of factors. With this dataset, we applied multidimensional linear regression to identify a function that captures the regular patterns associating motivation with its influencing factors. This prediction function quantifies a student's motivation – our target variable – based on the values of the independent variables. Using this function and the Monte Carlo method, we explored the multidimensional space of independent variables to estimate the probability that a student attains one of ten motivation levels, ranging from completely demotivated to highly motivated to learn scientific computing. Our findings indicate that students are most likely to achieve moderate motivation levels, specifically the 4th (26.86%), 5th (45.03%), and 6th (21.21%) levels. However, we also found that implementing effective policies and strategies (e.g., enhancing student satisfaction) may increase the probability of achieving higher motivation levels, particularly the 7th (36.98%) and 8th (61.04%) levels.**

Keywords-higher education; motivation; regression; Monte Carlo method.

I. Introduction

The motivation to study in college is crucial for success in courses. Demotivated students often lack the willingness, reasons, or desire to complete assignments, study for tests, and so forth. In the specific context of scientific computing courses, maintaining students' motivation can be challenging due to the difficulties associated with the mathematical concepts underlying these courses and understanding how to apply these concepts and methods to solve real-world problems.

Scientific computing courses, such as numerical methods, can be quite challenging to learn. This has prompted research aimed at predicting which students are at risk of failing these courses [1]–[4]. Indeed, these courses are difficult because they involve mathematics, programming skills, and knowledge of science for application purposes.

Studying the factors influencing the learning of mathematics has been a subject of interest in prior research, from basic educational levels [5]–[8] to higher education [9]–[13], and even doctoral levels [14]. Scientific computing, essentially an applied mathematics topic, consists of numerical methods and heuristics for solving mathematical problems in science and engineering that cannot be addressed analytically.

In Colombia, studies have examined the process of constructing knowledge among college students in the context of algebra courses within engineering curricula [10]. However, prior research has focused primarily on commitment, satisfaction, and the challenges of learning mathematics in college. As far as we know, no prior research has studied the students' motivation to learn scientific computing.

Thus, we aim to fill this gap in the literature. Our problem is to find a function that allows us to predict the student's motivation based on the variables that affect it by examining regular patterns in students' perceptions. Furthermore, we aim to explore a broader set of values for those *independent variables* than those available in the surveyed students' perceptions to calculate the probability that a student feels completely demotivated (level one) up to completely motivated (level ten). Identifying these probabilities is useful for developing policies that improve motivation among students pursuing careers that require scientific computing.

We found that it is less likely for a student to reach the highest level of motivation, while the fifth level is the most probable. Moreover, by simulating other scenarios, we found that it is possible to increase the student's motivation for learning scientific computing.

The remainder of this article is outlined as follows: In Section II, we describe how we collected and preprocessed the dataset, including its characteristics. Section III presents the regression model used to determine the functional relationship between the independent and dependent variables of this study, which forms the basis for calculating the probability of a student reaching a specific motivation level in scientific computing courses, as described in Section IV. To visualize the input variables of the regression model, we reduce the dimensionality of the input space and discuss the method used in Section V. Section VI presents and discusses our findings, while Section VII concludes the article and outlines directions for future research.

II. Collecting and Preprocessing the Dataset

To determine the aforementioned functional relationship, we collected a dataset to fit a regression model. The dataset

contains independent and dependent variables. The *independent variables*, also referred to as *input variables*, were selected from *factors* that influence students' motivation for learning scientific computing. The dependent variable, also known as the target or output variable, represents the student's level of motivation.

In this study, we assumed that the factors listed in Table I influence student motivation in scientific computing. Some of these factors have been utilized in prior research [12], [13].

We conducted a survey of 117 engineering students enrolled in scientific computing courses in 2024, specifically numerical methods and nonlinear programming. The identities of the students were anonymized. Each student ranked almost all factors on a scale from one to five, with the exception of the average grade in previous mathematics courses, which is represented as a real number between zero and five (inclusive). For example, the extent to which a student felt positively about the course ranged from "not good at all" (zero) to "completely positive" (five). In contrast, the target variable was measured on a scale from zero to ten.

After collecting the dataset, we performed an F-test to select the input variables used for fitting the regression model. Input variables with a p-value less than 5×10^{-2} were selected, rejecting the null hypothesis that there is no linear relationship between the input variable and the target variable. Table I lists the input variables selected according to this criterion.

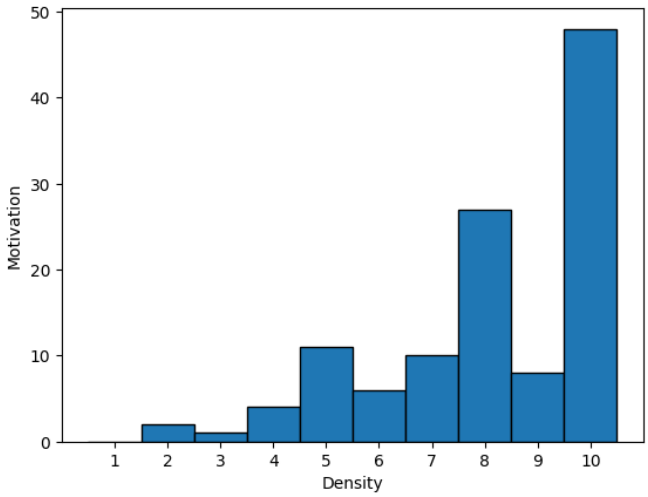


Figure 1. This histogram depicts the frequency with the students chose every level of motivation during the survey

Thus, the resulting dataset comprises 15 input variables in addition to the target variable. Each input variable forms a component of 117 vectors in a multidimensional input space, where each vector represents a student's responses to the survey, and the motivation level is recorded as the target variable. Formally, the vector $x_i \in \mathcal{X} \subset \mathbb{R}^D$ corresponds to the ith student, where the jth component x_{ij} represents the input variable associated with a specific factor. Here, D denotes the number of dimensions in the input space, i.e., $D = 15$. Additionally, the target variable $y_i \in \mathcal{Y} \subset \mathbb{R}$ represents the jth student's course motivation level.

Finally, the dataset is denoted as

$$\mathcal{D} = \{(x_i, y_i) \,|\, x_i \in \mathcal{X} \wedge y_i \in \mathcal{Y},\, i = 1, \ldots, N\} = \{(x_i, y_i)\}_{i=1}^{N},$$

where N is the number of examples in the dataset ($N = 117$), and the input vector space $\mathcal{X}$ is the Cartesian product of the domains of each input variable, i.e.,

$$\mathcal{X} = \mathbb{X}_1 \times \mathbb{X}_2 \times \cdots \times \mathbb{X}_j \times \cdots \times \mathbb{X}_D.$$

The domain of the j-th input variable is defined as

$$\mathbb{X}_j = \{a \in \mathbb{N} \,|\, 1 \leq a \leq 5\}, \quad \text{for } j = 1, \ldots, D.$$

On the other hand, the domain of the target variable is denoted as

$$\mathcal{Y} = \{a \in \mathbb{N} \,|\, 1 \leq a \leq 10\}.$$

The histogram, shown in Figure 1, illustrates that the maximum motivation level was chosen by most of the students, namely, 48 out of 117 students (see Table II).

III. Finding the Functional Relation Among Variables

The functional relationship between the target variable and the input variables is determined using *ridge regression* (a more detailed description of this method can be found in [15]). The goal is to find the function g based on the dataset $\mathcal{D}$, such that it approximates the target variable y_i given the input variables in the vector x_i, i.e., $g(x_i) \approx y_i$, where $g : \mathcal{X} \to \mathcal{Y}$. The function g is defined as a linear combination of weights and input variables as follows:

$$g(x_i) = w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_D x_{iD} = w^T \hat{x}_i,$$

where $\hat{x}_i$ is an augmented vector of x_i that includes an additional component $\hat{x}_{i0} = 1$, and $\hat{x}_{ij} = x_{ij}$ for $j = 1, \ldots, D$. The components of the vector $w \in \mathbb{R}^{D+1}$ are the weights of the model. Henceforth, we shall call g the *prediction function*, as it allows us to predict a student's motivation level given the above-mentioned input variables.

For simplicity sake, the input variables are represented by the matrix $X \in \mathbb{R}^{N \times (D+1)}$, where $X_{ij} = 1$ if $j = 1$, and $X_{ij} = \hat{x}_{i,j-1}$ for $j = 2, \ldots, D+1$. Let $y \in \mathbb{R}^N$ be the vector whose components correspond to the target variable values in the dataset. This setup formalizes the problem as the following optimization problem:

$$\min_w f(w) = ||Xw - y||^2 + \lambda ||w||^2, \tag{1}$$

where λ is the *regularization parameter* used to prevent *overfitting* to noise in the input variables. The regularization parameter is chosen using the *elbow rule* and *10-fold cross-validation*. By setting the gradient of the objective function f with respect to w to zero and rearranging terms, the solution is obtained as:

$$w = (X^T X + \lambda I)^{-1} X^T y. \tag{2}$$

TABLE I. INPUT VARIABLES ASSOCIATED TO THE FACTORS THAT INFLUENCE THE STUDENT'S MOTIVATION IN SCIENTIFIC COMPUTING COURSES

Input Variable	*F-statistic*	*p-value*
The student's average grade in previous mathematics courses	0.43	5.16×10^{-1}
The extent to which the student has felt good about the course† $x_{i,1}$	26.17	1.27×10^{-6}
The extent to which the student has felt good about previous mathematics courses† $x_{i,2}$	24.68	2.38×10^{-6}
The extent to which the student has enjoyed the course† $x_{i,3}$	37.08	1.54×10^{-8}
The extent to which the student considers it imperative to study the course	1.08	3.02×10^{-1}
The extent to which the student considers it imperative to study mathematics courses	0.99	3.22×10^{-1}
The extent to which the student considers it wrong not to study the course	0.40	5.26×10^{-1}
The extent to which the student considers it wrong not to study mathematics courses	1.30	2.56×10^{-1}
The extent to which the student would like to recommend the course to other peers† $x_{i,4}$	37.27	1.43×10^{-8}
The extent to which the student perceives the university has up-to-date equipment† $x_{i,5}$	8.43	4.42×10^{-3}
The extent to which the course has been encouraged students to study with classmates† $x_{i,6}$	29.49	3.17×10^{-7}
The extent to which the student has been encouraged to help classmates† $x_{i,7}$	29.09	3.74×10^{-7}
The extent of the student's current engagement in participating in course lessons† $x_{i,8}$	20.43	1.51×10^{-5}
The extent of the student's current engagement in attending course lessons	2.04	1.56×10^{-1}
The extent of the student's current engagement in making an additional effort to understand the course† $x_{i,9}$	27.31	7.82×10^{-7}
The extent of the student's current focus and engagement during course lessons† $x_{i,10}$	27.59	6.96×10^{-7}
The extent to which the student has been encouraged to study the course independently† $x_{i,11}$	31.37	1.48×10^{-7}
The extent to which the student has believed the course is useful for their professional life† $x_{i,12}$	20.64	1.37×10^{-5}
The extent to which the student has considered mathematics courses useful for their professional life† $x_{i,13}$	12.94	4.75×10^{-4}
The extent to which the student has believed that they possess the ability to learn mathematics† $x_{i,14}$	3.30	7.17×10^{-2}
The extent to which the student has believed that they have the ability to solve mathematics-related problems	0.93	3.37×10^{-1}
The extent to which the student has enjoyed to solve challenging mathematics-related problems similar to those addressed in the course	15.02	1.77×10^{-4}
The extent to which the student feels their secondary school preparation is insufficient for succeeding in mathematics courses	3.67	5.79×10^{-2}
The extent to which the student believes people have innate abilities for mathematics	1.96	1.64×10^{-1}
The extent to which the student believes learning success depends on the lecturer	3.80	5.36×10^{-2}
The extent to which the student believes learning success depends on the student	1.62	2.06×10^{-1}
The extent to which the student believes hard work is key to succeeding in the course† $x_{i,15}$	4.29	4.07×10^{-2}

†The input variable is selected for regression

TABLE II. MOTIVATION LEVELS OF THE STUDENTS WHO ANSWERED THE SURVEY

Motivation Level	*Number of Students*	*Proportion of the Sample*
2	2	1.71%
3	1	0.85%
4	4	3.42%
5	11	9.40%
6	6	5.13%
7	10	8.55%
8	27	23.08%
9	8	6.84%
10	48	41.02%
Total	117	100.00%

Once the weights are computed, the function that maps the input variables to the target variable is defined. Thus, given new input variables x_{new} corresponding to a new student, the function $g(x_{\text{new}})$ calculates their respective motivation level.

IV. CALCULATING THE PROBABILITY OF EACH MOTIVATION LEVEL

In this section, we delve into the details of calculating the probability that students feel motivated at each level using the function that predicts the target variable given the input variables. To achieve this, we adopted the Monte Carlo numerical method [16].

The probability that students achieve motivation level k for learning scientific computing is defined as follows:

$$P(y_i = k) \approx P(g(x_i) = k) = \int_{\mathcal{X}} P(g(x_i) = k \,|\, x_i) P(x_i)\, dx_i, \tag{3}$$

where $P(x_i)$ is the probability density function of the input variables.

Assuming that each component of x_i is uniformly distributed, i.e., $x_{ij} \sim \mathcal{U}(1, 5)$ for $j = 1, \ldots, D$, the probability density function $P(x_i)$ is uniform. Therefore, Equation (3) is rewritten as:

$$P(y_i = k) \approx P(g(x_i) = k) \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(g(x_i) = k), \tag{4}$$

where N is the number of vectors x_i, whose components are random numbers uniformly distributed. Moreover, $\mathbf{1}(u) = 1$ if u is true, and $\mathbf{1}(u) = 0$ otherwise. Note that N is not the size of the dataset described in Section II.

The value of N is chosen based on the standard error (SE), which is calculated as:

$$SE = \frac{\sigma}{\sqrt{N}}, \tag{5}$$

where σ is the standard deviation of the calculated probabilities. The value of N is increased iteratively until the SE decreases to a tolerable threshold.

V. Reducing the Dimensionality of the Input Space

We reduce the dimensionality of the vectors in the input space by adopting Principal Component Analysis (PCA) (see [15] for a complete description of the method). PCA transforms the original vectors into a new space with fewer dimensions, where each principal component is a linear combination of the original variables. The components are calculated to maximize variance, enabling a better understanding of the latent structure, reducing noise, and visualizing the dataset.

Let $u_1 \in \mathbb{R}^D$ be the first basis vector of the new space. The first principal component z_{i1}, corresponding to vector x_i, is calculated as:

$$z_{i1} = u_1^T x_i, \tag{6}$$

where the variance of the first principal component is defined as:

$$\text{var}(z_{i1}) = u_1^T S u_1, \tag{7}$$

with $S \in \mathbb{R}^{D \times D}$ being the covariance matrix:

$$S = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T. \tag{8}$$

Here, $\bar{x}$ is the mean of the input vectors.

The optimization problem is to find u_1 that maximizes the variance of the first principal component z_{i1}, subject to the constraint that u_1 is a unit vector ($u_1^T u_1 = 1$):

$$\max_{u_1} J(u_1) = u_1^T S u_1 - \lambda_1 (u_1^T u_1 - 1), \tag{9}$$

where λ_1 is a Lagrange multiplier. By setting the gradient of the objective function J to zero with respect to u_1 and simplifying, we find:

$$\lambda_1 = u_1^T S u_1 = \text{var}(z_{i1}), \tag{10}$$

where λ_1 represents the variance of the first principal component.

To calculate the second principal component z_{i2}, we find the second basis vector u_2, such that $z_{i2} = u_2^T x_i$. The vector u_2 must be orthogonal to u_1 ($u_1^T u_2 = 0$) and a unit vector ($u_2^T u_2 = 1$). The optimization problem is:

$$\max_{u_2} J(u_2) = u_2^T S u_2 - \lambda_2 (u_2^T u_2 - 1) - \alpha u_2^T u_1, \tag{11}$$

where λ_2 and α are Lagrange multipliers. Setting the gradient of J to zero yields:

$$\lambda_2 = u_2^T S u_2 = \text{var}(z_{i2}). \tag{12}$$

This procedure generalizes to calculate all basis vectors u_j for $j = 1, \ldots, D$, producing d principal components, where $d < D$. The transformed vector $z_i \in \mathbb{R}^d$ represents the original $x_i \in \mathbb{R}^D$ in the reduced space. The vector z_i can then be used as input for the methods described in Sections III and IV.

For visualization purposes, three or fewer principal components are sufficient ($d \leq 3$). Another criterion for choosing d is based on the proportion of retained variance ρ, calculated as:

$$\rho = 100 \cdot \frac{\sum_{j=1}^{d} \lambda_j}{\sum_{k=1}^{D} \lambda_k} \%. \tag{13}$$

In some cases, $\rho = 100\%$ is achieved with $d < D$ due to noise in the variables. Alternatively, a threshold for ρ (e.g., $\rho \leq 97\%$) can be set depending on the application domain.

VI. Results and Discussion

The resulting prediction function g estimated through the regression model described in Section III is defined as follows:

$$\begin{aligned} g(x_i) = {} & 0.0220 + 0.1678x_{i,1} + 0.1751x_{i,2} + 0.1992x_{i,3} + \ldots \\ & \cdots + 0.1989x_{i,4} + 0.1018x_{i,5} + 0.1111x_{i,6} + \ldots \\ & \cdots + 0.1592x_{i,7} + 0.1157x_{i,8} + 0.1597x_{i,9} + \ldots \\ & \cdots + 0.1557x_{i,10} + 0.1765x_{i,11} + \ldots \\ & \cdots + 0.0895x_{i,12} - 0.0049x_{i,13} + \ldots \\ & \cdots + 0.0744x_{i,14} + 0.0749x_{i,15} \end{aligned} \tag{14}$$

According to this model, the ith student's satisfaction ($x_{i,4}$) and enjoyment ($x_{i,3}$) with the scientific computing course are the variables with the highest weights; therefore, both correspond to the most influential factors in the student's motivation for the course.

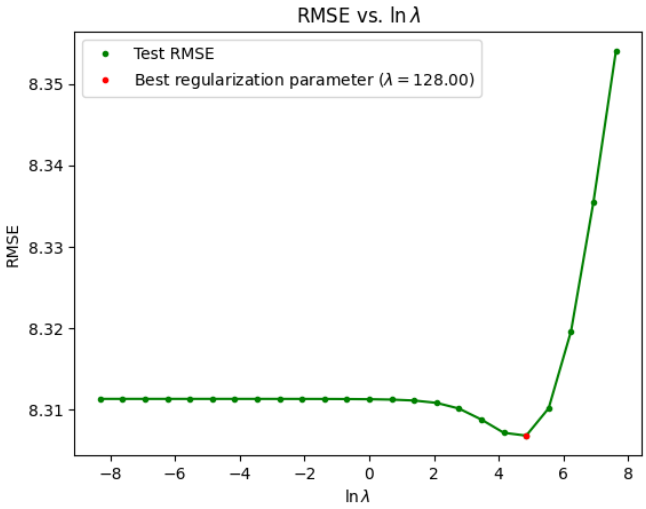


Figure 2. This chart is the elbow rule used to tune the regularization parameter of the regression model. We used values of λ ranging from 2^{-12} to 2^{12}. We adopted 10-fold cross-validation to evaluate the model with several regularization settings.

On the other hand, a negative weight for $x_{i,13}$ indicates that students who perceive mathematics courses as more useful for their careers tend to have slightly lower motivation levels in scientific computing courses. Since the absolute value of the weight is very small, the effect is weak but still worth considering. Either the students are motivated to obtain high grades in the scientific computing course, even though they consider acquiring mathematical knowledge merely a graduation requirement, or they are motivated because they can solve mathematical problems in the scientific computing course using algorithms (i.e., numerical methods or heuristics) instead of the analytical methods covered in previous classical courses (e.g., differential calculus).

The regression model achieved a coefficient of determination and a root-mean-squared-error of 0.37 and 1.62, respectively. This outcome suggests that the regression model performs better than predicting by using the mean value. The regularization parameter λ was chosen using the elbow rule as depicted in Figure 2. According to this method, the best value for regularization is $\lambda = 128$.

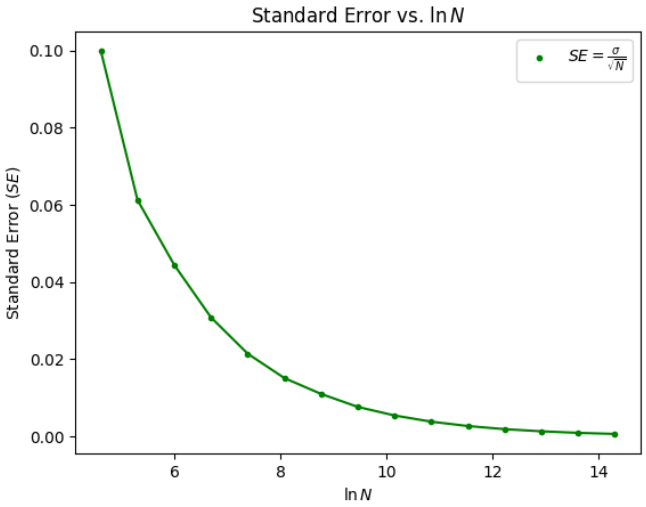


Figure 3. This chart shows how the standard error drops as the variable N is increased.

Thus, the maximum and minimum values that can be obtained from the prediction function are approximately 2 and 10, respectively. However, the most probable level according to the Monte Carlo method is 4.908 with a standard error of 6.8×10^{-4}. Figure 3 illustrates how this value was achieved as N increases. This outcome was obtained with 95% confidence (alpha = 0.05), within the interval (4.90, 4.91). However, this is not an actual level; therefore, we performed rounding to the nearest even number for halfway cases, and the probabilities of the motivation levels are shown in Table III.

It is noteworthy that the high levels of motivation have the lowest probabilities, in contrast with the motivation levels of the students in the dataset depicted in Table II. The results in Table III align with the histogram illustrated in Figure 4, where the high levels were obtained with less frequency through the Monte Carlo simulation.

TABLE III. PROBABILITY OF EVERY MOTIVATION LEVEL CALCULATED WITH THE MONTE CARLO METHOD

Level	*Probability*
1	P(y = 1.0) = 5.49×10^{-4}%
2	P(y = 2.0) = 1.34×10^{-1}%
3	P(y = 3.0) = 4.17%
4	P(y = 4.0) = 26.86%
5	P(y = 5.0) = 45.03%
6	P(y = 6.0) = 21.21%
7	P(y = 7.0) = 2.55%
8	P(y = 8.0) = 5.57×10^{-2}%
9	P(y = 9.0) = 6.10×10^{-5}%

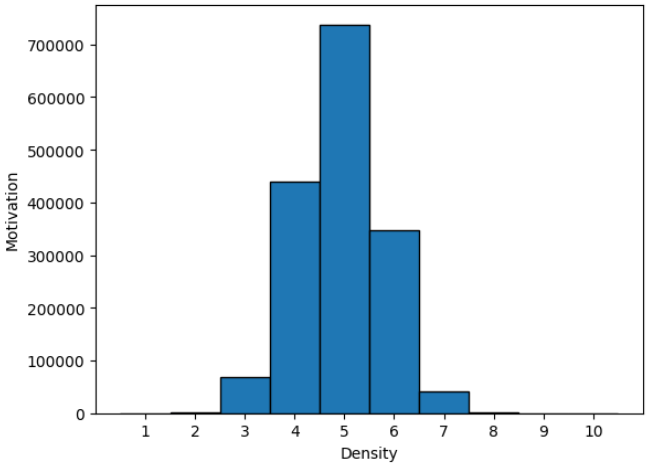


Figure 4. Histogram yielded through the Monte Carlo method. This shows the frequency with which the function g calculates each motivation level based on the random input variables.

Despite these figures, we might increase the probability of high motivation levels through policies that improve the factors associated with the input variables. We found this by simulating $P(y_i = k)$ while assuming high random values for the controllable input variables, such as $x_{ij} \sim \mathcal{U}(3, 5)$ for $1 \leq j \leq 12$, and setting $x_{i,12}$ to zero to mitigate its negative influence on the prediction. Meanwhile, the variables that cannot be directly controlled were assigned random values across their entire range, i.e., $x_{ij} \sim \mathcal{U}(1, 5)$ for $j = 13, 14$.

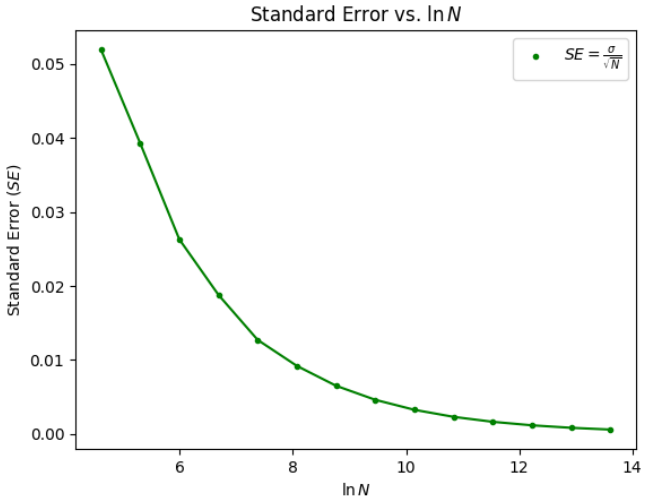


Figure 5. This chart shows how the simulation converges to the solution with a setting aiming to increase the student motivation to learn scientific computing, hence, standard error drops as the variable N is increased.

With the aforementioned simulation setting the most probable level according to the Monte Carlo method is 7.642 with a standard error of 8.1×10^{-4}. Figure 5 shows how the simulation converges as N increases. This outcome was obtained with 95% confidence (alpha = 0.05), within the interval (7.641,

7.643). The probabilities of the motivation levels are shown in Table IV while the resulting histogram is illustrated in Figure 6.

TABLE IV. PROBABILITY OF EVERY MOTIVATION LEVEL CALCULATED WITH THE MONTE CARLO METHOD FROM THE BEST SIMULATION SETTING

Level	*Probability*
6	P(y = 6.0) = 2.5×10^{-1}%
7	P(y = 7.0) = 36.98%
8	P(y = 8.0) = 61.03%
9	P(y = 9.0) = 1.74%

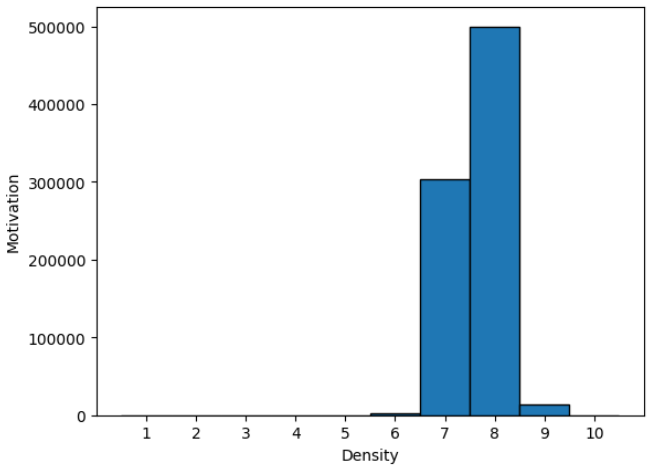


Figure 6. Histogram yielded through the Monte Carlo method using a simulation setting that aims to increase the probability of highest level of motivation for learning scientific computing

TABLE V. VARIANCE RETAINED BY THE PRINCIPAL COMPONENTS

Number of Principal Components	*Retained Variance (%)*
1	44.84%
2	55.43%
3	64.22%
4	71.16%
5	76.19%
6	80.51%
7	84.30%
8	87.74%
9	90.69%
10	92.92%
11	94.77%
12	96.50%
13	97.99%
14	99.15%
15	100.00%

The previous results reveal that designing policies to enhance satisfaction in prerequisite mathematics and scientific computing courses, providing students with up-to-date equipment, promoting teamwork, and so forth might improve students' motivation to learn scientific computing.

Additionally, regarding the visualization of the input space, using two components retained 53.43% of the variance (see Table V). To retain the complete variance, the dimensionality must be the same as the original input space or close to it. Therefore, it is pointless to use a dimensionality that cannot be visualized.

The regression model trained with a two-dimensional input space obtained through principal component analysis yields a coefficient of determination and a root-mean-squared-error of 0.33 and 1.67, respectively. Therefore, the regression model applied to the original input space outperforms the one applied to the reduced input space.

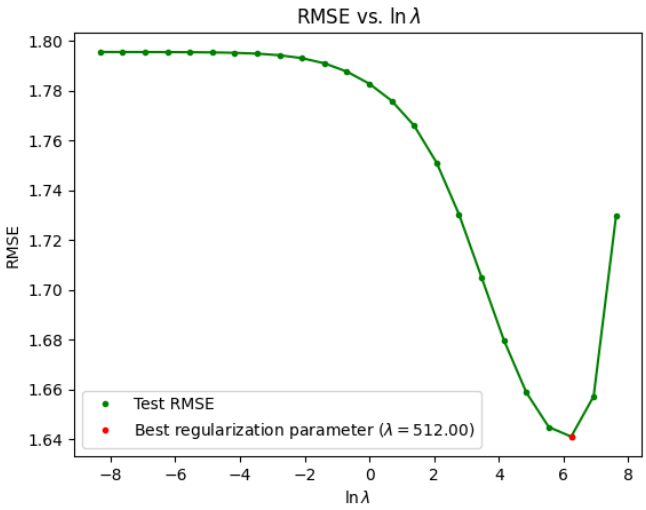


Figure 7. This chart is the elbow rule used to tune the regularization parameter of the regression model performed on a two-dimensional input space. We used values of λ ranging from 2^{-12} to 2^{12}. We utilized 10-fold cross-validation to evaluate the model with different regularization values.

Figure 7 shows how the regularization parameter was selected through the elbow rule to avoid overfitting.

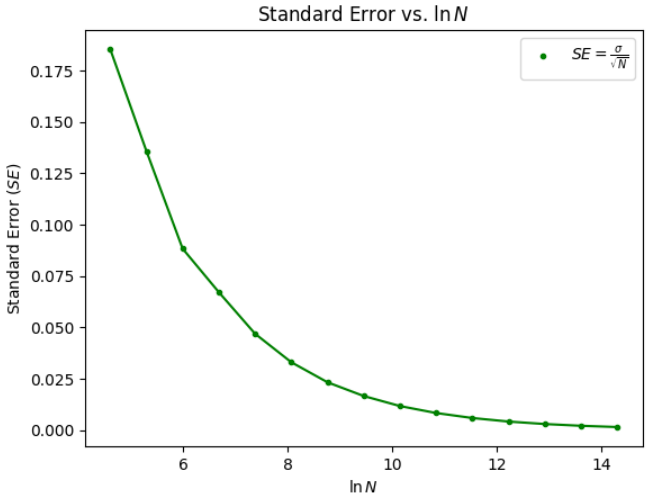


Figure 8. This chart shows how the standard error drops as the variable N is increased in the Monte Carlo simulation applied on the two-dimensional input space.

When the Monte Carlo method is performed on the regression model obtained from the new two-dimensional input space,

the outcome is that the most likely level is 3.84 with a standard error of 1.46×10^{-3}. This outcome is within (3.83, 3.84) with a 95% (alpha = 0.05) confidence interval (see Figure 8)

TABLE VI. PROBABILITY OF EVERY MOTIVATION LEVEL CALCULATED WITH THE MONTE CARLO METHOD TAKING INTO ACCOUNT TWO PRINCIPAL COMPONENTS

Level	*Probability*
1	P(y = 1.0) = 13.62%
2	P(y = 2.0) = 15.61%
3	P(y = 3.0) = 15.64%
4	P(y = 4.0) = 15.59%
5	P(y = 5.0) = 15.62%
6	P(y = 6.0) = 15.59%
7	P(y = 7.0) = 8.32%

The results obtained with two-dimensional input spaces reveal that it becomes more likely that a student will reach the lower levels of motivation (see Table VI). The results align with the histogram depicted in Figure 9. In this case, the prediction function is defined as follows:

$$g(z_i) = 0.03 + 0.52z_{i1} - 0.04z_{i2} \quad (15)$$

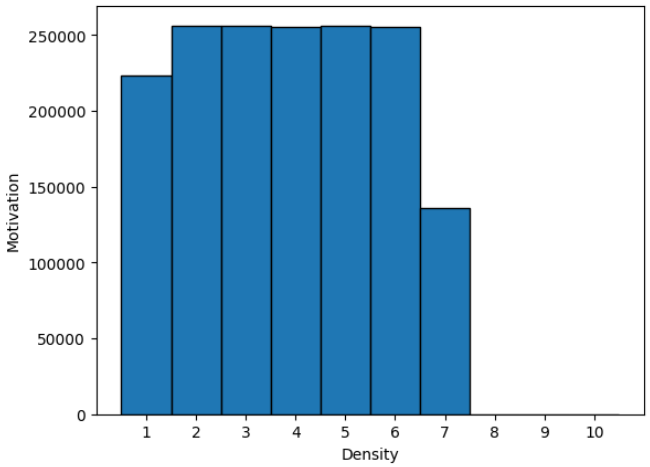


Figure 9. Histogram yielded through the Monte Carlo method on the two-dimensional input space. This shows the frequency of each motivation level based on the reduced input space.

Figure 10 shows that the second principal component, z_{i2}, is mostly negative, despite its corresponding weight also being negative (see Equation (15)). Moreover, this principal component does not appear to be a latent factor influencing student motivation, whereas the first principal component does. This is evident from the figure, as the vectors on the left side correspond to the highest motivation levels, while those on the opposite side are associated with the lowest levels.

In Figure 10, the vectors representing the students who participated in the survey are shown, classified according to the function g obtained from the regression model, as illustrated by the contour lines. Notably, these lines indicate the absence of vectors in the highest motivation level, which explains why this level is unlikely in the simulation outcomes.

VII. CONCLUSION AND FUTURE WORK

In this study, we found that engineering students enrolled in scientific computing courses at the University of Córdoba exhibit a moderate level of motivation, which is the most likely outcome. This suggests that these students find it challenging to grasp the concepts, foundations, and methods taught in these courses. As such, understanding the underlying factors that contribute to motivation is critical in designing more effective learning environments.

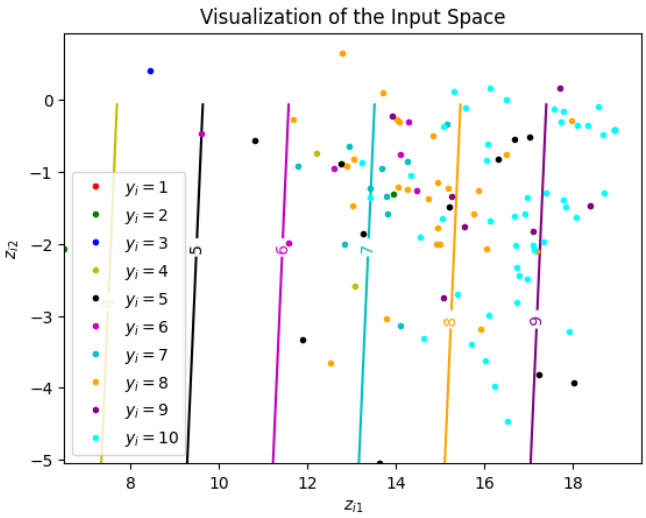


Figure 10. Visualization of the latent factors derived from the regression model. The contour lines show the lower probability of obtaining the higher motivation levels.

As a consequence, it is essential for lecturers to develop effective motivation strategies tailored to the unique challenges of scientific computing courses. Strategies might include more interactive teaching methods, real-world problem-solving tasks, or personalized feedback systems aimed at increasing student engagement. By aligning teaching practices with the motivational needs of students, there is a greater chance of improving learning outcomes and overall academic success.

However, the findings are subject to some limitations. The data collected in this study might not fully capture the diversity of motivation levels across different institutions or disciplines. Therefore, further data collection is necessary to address potential validity threats, such as the potential for selection bias or the lack of data from different academic backgrounds. Expanding the dataset to include students from other universities or fields of study could provide a more comprehensive understanding of the factors influencing student motivation.

So far, to interpret the prediction function, we have assumed a linear relationship between the factors influencing motivation and the student's motivation level. While this assumption has provided valuable insights, it might not fully reflect the complexity of motivation. For future research, we will explore nonlinear regression models such as Gaussian processes, Kernel

Ridge Regression, and Random Forests, which might uncover more nuanced relationships between the variables.

Additionally, we will also evaluate alternative models for dimensionality reduction. Methods, such as non-negative matrix factorization, autoencoders, and t-SNE might be adopted to better capture the underlying structure of the data. These methods may offer advantages over principal component analysis, particularly in terms of capturing nonlinearities or latent factors that influence motivation.

The impact of further data collection should not be underestimated. By collecting more data, we could improve the robustness of our findings and potentially develop a model that can predict motivation levels more accurately across diverse student populations. Future work could also involve incorporating more detailed demographic information, which could lead to insights into how motivation varies across different student groups based on age, prior experience, or other factors.

Finally, future work will include practical steps to further investigate these findings. For example, we plan to collaborate with faculty members in scientific computing courses to apply the insights gained from this study in real-world teaching settings. Pilot studies may also be conducted to test the efficacy of the proposed motivational strategies and validate the results through student feedback and performance.

In summary, while this study provides a valuable starting point, there is much more to explore regarding the complexities of student motivation in scientific computing. We hope that future research will contribute to the development of more effective and personalized teaching methods that foster greater student engagement and success.

ACKNOWLEDGMENT

Caicedo-Castro thanks the Lord Jesus Christ for blessing this project. The authors thank Universidad de Córdoba in Colombia for supporting this study. They also thanks all students who collaborated by answering the survey conducted for collecting the dataset used in this study. Finally, the author thanks the anonymous reviewers for their comments that contributed to improve the quality of this article.

REFERENCES

[1] I. Caicedo-Castro, M. Macea-Anaya, and S. Rivera-Castaño, "Early Forecasting of At-Risk Students of Failing or Dropping Out of a Bachelor's Course Given Their Academic History - The Case Study of Numerical Methods," in *PATTERNS 2023: The Fifteenth International Conference on Pervasive Patterns and Applications*, ser. International Conferences on Pervasive Patterns and Applications. IARIA: International Academy, Research, and Industry Association, 2023, pp. 40–51.

[2] I. Caicedo-Castro, "Course Prophet: A System for Predicting Course Failures with Machine Learning: A Numerical Methods Case Study," *Sustainability*, vol. 15, no. 18, 2023, 13950.

[3] ——, "Quantum Course Prophet: Quantum Machine Learning for Predicting Course Failures: A Case Study on Numerical Methods," in *Learning and Collaboration Technologies*, P. Zaphiris and A. Ioannou, Eds. Cham: Springer Nature Switzerland, 2024, pp. 220–240.

[4] ——, "An Empirical Study of Machine Learning for Course Failure Prediction: A Case Study in Numerical Methods," *International Journal on Advances in Intelligent Systems*, vol. 17, no. 1 and 2, pp. 25–37, 2024.

[5] L. Ayebale, G. Habaasa, and S. Tweheyo, "Factors Affecting Students' Achievement in Mathematics in Secondary Schools in Developing countries: A Rapid Systematic Review," *Statistical Journal of the IAOS*, vol. 36, pp. 1–4, 2020.

[6] M. Gómez-García, H. Hossein-Mohand, J. M. Trujillo-Torres, H. Hossein-Mohand, and I. Aznar-Díaz, "Technological Factors That Influence the Mathematics Performance of Secondary School Students," *Mathematics*, vol. 8, no. 11, 2020, 1935.

[7] J.-M. Trujillo-Torres, H. Hossein-Mohand, M. Gómez-García, H. Hossein-Mohand, and F.-J. Hinojo-Lucena, "Estimating the Academic Performance of Secondary Education Mathematics Students: A Gain Lift Predictive Model," *Mathematics*, vol. 8, no. 12, 2020, 2101.

[8] M. Maamin, S. M. Maat, and Z. H. Iksan, "The Influence of Student Engagement on Mathematical Achievement among Secondary School Students," *Mathematics*, vol. 10, no. 1, 2022, 41.

[9] A. Brezavšček, J. Jerebic, G. Rus, and A. Žnidaršič, "Factors Influencing Mathematics Achievement of University Students of Social Sciences," *Mathematics*, vol. 8, no. 12, 2020, 2134.

[10] E. Martinez-Villarraga, I. Lopez-Cobo, D. Becerra-Alonso, and F. Fernández-Navarro, "Characterizing Mathematics Learning in Colombian Higher Distance Education," *Mathematics*, vol. 9, no. 15, 2021, 1740.

[11] J. Park, S. Kim, and B. Jang, "Analysis of Psychological Factors Influencing Mathematical Achievement and Machine Learning Classification," *Mathematics*, vol. 11, no. 15, 2023, 3380.

[12] S. Batista-Toledo and D. Gavilan, "Student Experience, Satisfaction and Commitment in Blended Learning: A Structural Equation Modelling Approach," *Mathematics*, vol. 11, no. 3, 2023, 749.

[13] M. Charalambides, R. Panaoura, E. Tsolaki, and S. Pericleous, "First Year Engineering Students' Difficulties with Math Courses- What Is the Starting Point for Academic Teachers?" *Education Sciences*, vol. 13, no. 8, 2023, 835.

[14] T. T. Wijaya, B. Yu, F. Xu, Z. Yuan, and M. Mailizar, "Analysis of factors affecting academic performance of mathematics education doctoral students: A structural equation modeling approach," *International Journal of Environmental Research and Public Health*, vol. 20, no. 5, 2023, 4518.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[16] N. Metropolis and S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

On the Operationalization of Composable Architecture by Means of Normalized Systems Theory

Geert Haerens
Antwerp Management School, Belgium
Engie nv, Belgium
Email: geert.haerens@engie.com

Herwig Mannaert
University of Antwerp, Belgium
Email: herwig.mannaert@uantwerpen.be

***Abstract*—In a fast-evolving world, companies require IT solutions that allow them to adapt swiftly to changing conditions. In 2020, Gartner introduced the Composable Architecture Framework as a guiding principle to create application landscapes that are easily composable and re-composable, thus supporting change. The Normalized Systems theory is about the creation of evolvable modular software. The concepts presented in the Composable Architecture Framework resonate with Normalized Systems. A closer analysis of the framework through Normalized Systems theory reveals that Gartner's framework lacks precision. As such, following the guidance of the framework will insufficiently protect companies from change, both outside and inside the organization.**

Keywords*—*Normalized Systems Theory; Composable Architecture; Packaged Business Capabilities.

I. Introduction

For many years now, a death wish toward the monolithic application has been declared. Monolithic applications are difficult to change and unsuitable in our fast-moving world. Many paradigms have been proposed over the years that aim at splitting applications into smaller, modular parts. With the rise of the Internet and faster network speeds, the physical distribution of those parts has become a reality. Distributed Computing Environment (DCE) [1] proposed modules encapsulating functionality that could be activated using Remote Procedure Calls (RPC). An essential benefit to splitting applications into smaller, independent callable modules is reuse. This resonates with McIlroy's dream, expressed during the 1968 NATO conference on Sofware Engineering, where *" ... I expect families of routines to be constructed on rational principles so that families fit together as building blocks. In short, [the user] should be able to safely regard components as black boxes."*. SAP ERP (Enterprise Resource Planning), for instance, uses this kind of approach to allow the calling of SAP functionalities from other systems through Remote Function Calls (RFC). SAP re-baptized RFC to BAPI, Business Application Programming Interface, to focus even more on encapsulated functionality. In the past couple of years, we have seen a shift from encapsulated functionality toward technology, meaning that today, the talk of the town is about REST APIs, message queues, and event systems as a means of implementing integration between modules but without paying attention to the functionality.

In 2020, Gartner [2] introduced the notion of Packaged Business Capabilities (PBC) to create Composable Applications. They proposed the Composable Architecture Framework [2] as a guide for proper encapsulating functionalities and using technologies that allow flexibility in recomposition and evolution. Their approach connects the previous focus on functionality with today's emphasis on technology.

Normalized System theory (NS) [3], originating in software development, put forward the necessary conditions for the evolvability of modular structures. In this paper, we will try to operationalize and guide the implementation of the concepts of Gartner's Composable Architecture Framework by using NS [4]. The paper is structured as follows: In Section II, we will introduce Gartner's Reference Architecture for Composable Business Technology, followed by Section III that will introduce NS. In Section IV, we refer to related work, and in Section V, we attempt to operationalize the framework through NS. Section VI reflects on the operationalization and the paper is wrapped up in Section VII.

II. Gartner's Reference Architecture for Composable Business Technology

In 2020, the world was struck by COVID. Besides the human loss and suffering, businesses were also severely disrupted by this crisis. Gartner noticed that companies with a modular application approach could adjust swiftly to external conditions by quickly and safely assembling, disassembling and reassembling applications as the world required. In November 2020, Gartner published their Reference Architecture for Composable Business Technology [2], which investigates the conditions required for a platform to facilitate the composition and re-composition of applications. The needed ingredients for such a platform are PBCs as the essential modules, an application composition experience allowing custom assembly of PBCs via low code and no code, an application composition platform enabling development and deployment of the newly composed applications, and a data fabric allowing easy access to data and analytics on those data.

The PBCs are modules that encapsulate a well-defined business capability (recognized by the intended business users) and must adhere to the following conditions:

- They are modular and cohesive.
- They are autonomous (can run independently) and have minimal dependencies with external components.
- They allow orchestration as they can realize a process flow across PBCs (via APIs, events, etc.).

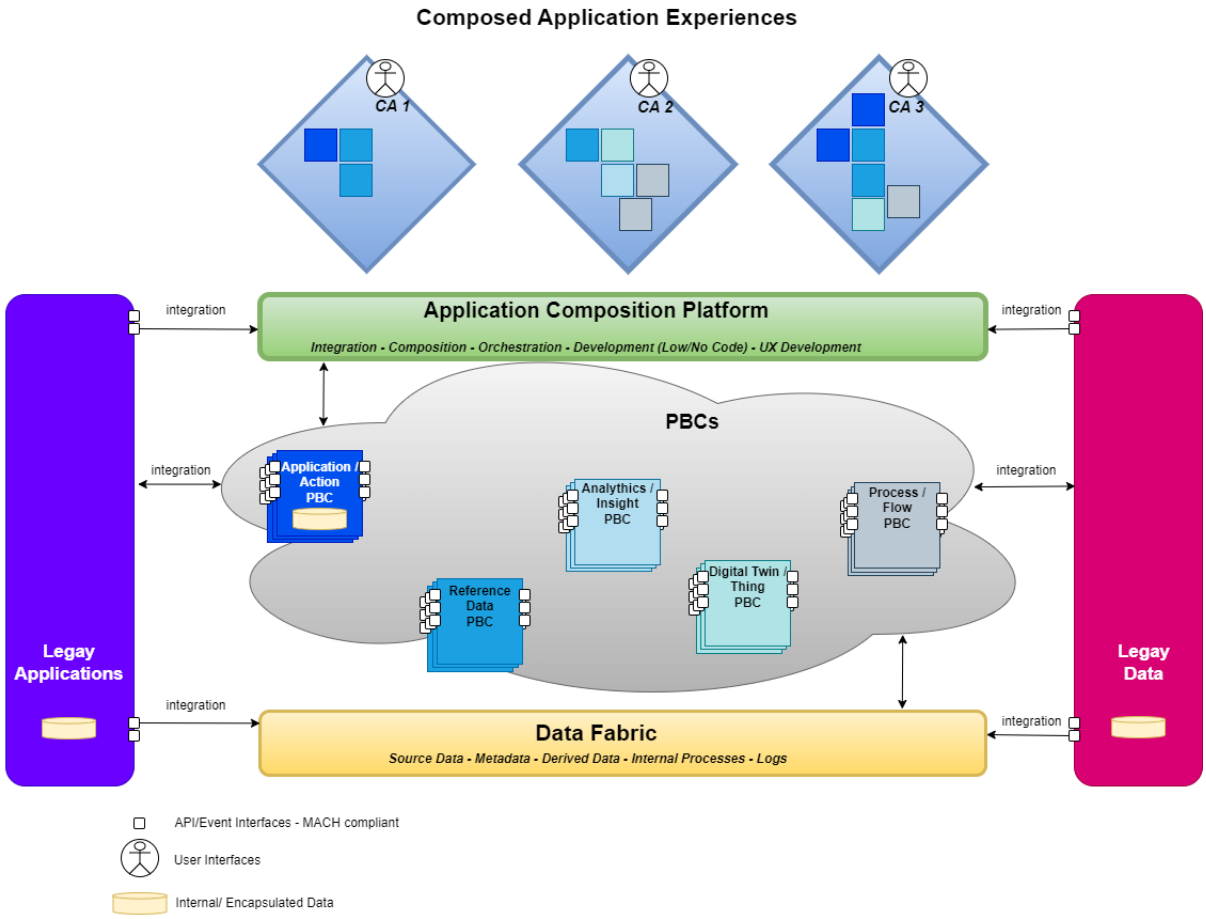


Fig. 1. Overview of the Composable Architecture Framework [5].

- They are discoverable as easily accessible and recognizable by those who require them.

Defining the right level of business capability granularity is challenging (too large = monolith, too small = more complex to identify).

Gartner provides further guidance on the PBC definition by defining PBC types.

There is the Application Type PBC that encapsulates both data and functions related to a well-defined business capability. Application Type PBC can be used to create fully expressed (=autonomous and encapsulating a full context) PBCs or basis business function PBC (= not autonomous and encapsulating a part of a context). Application Type PBC can create pseudo PBCs that are APIs toward existing monolithic applications. They encapsulate the legacy application, allowing them to participate in PBC composition but lack some flexibility in true PBCs regarding their evolvability.

Reference-type PBCs allow encapsulated access to data in the data fabric. They can access master data, metadata, or any data instance representing a business object or physical data container.

The Insight Type PBC allows the encapsulation of data analytics processes. It can perform analytic operations or apply AI models (ML, Deep learning, etc.) to data in the data fabric.

The Thing Type PBC encapsulates things in the physical world. It can be used to access and manipulate data from the real world (IoT).

There is the Flow Type PBC that combines different PBC in an order. Flow Type PBCs facilitate the creation of orchestrated PBCs that encapsulate a process.

The PBCs are activated via event channels and called via APIs. They can also provide different and optional user interfaces (web, mobile, or other).

In Table 1, Gartner contrasts the organization of an application landscape in PBCs (combined with a composition and deployment platform) with traditional application landscapes.

Garter analyzed the main change drivers in their papers: adding new business capabilities and their associated PBCs. Gartner further emphasizes that a platform that conforms to the above specifications is insufficient. Business and IT must work closely together to define and implement the required business capabilities, requiring what Gartner calls fusion teams. These teams are essential in creating business–IT alignment and, thus, value creation.

To start with a composable architecture strategy, one must first know what PBCs are already in the company. They may

TABLE I
CONTRASTING TRADITIONAL WITH PBC APPLICATION LANDSCAPES (FROM [2])

Criteria	Traditional Applications	PBCs
Primary value	Business capability	Business capability
Primary access	User Interface (UI)	Programmatic Interface (API, event)
Scope	Many business objects	One business object
Internal architecture	monolith or modular	monolith or modular
Designed for	Business	Business and IT
Design priority	Stability	Agility
Delivered value	Business solutions	Recomposable business solutions
Production style	Project	Product
Essential tools	Customization included	Composition, added cost
Required IT skills	Customization, low	Composition, high
Cost	Bulk, some ”shelfware”	Componentized, tracks value
Governance	Sample	Complex
Internal data	”Owned”	”Owned”
Open for integration/composition	Partially, a secondary priority	Fully, primary design objective

already be present as fine-grained PBCs or via applications aggregating PBCs and accessible via APIs. It is vital that investments in future technologies that should provide PBCs can be used as such. The individual PBCs must be accessible via APIs and not in an aggregated way. For example, a SaaS solution integrating multiple PBCs should allow the individual usage of the platform PBCs without depending on the other PBCs.

New applications are composed of assembled PBCs, allowing faster delivery and updating. Updating a PBC in the catalogue should update all applications with that PBC as an active module.

In summary, the Gartner Composable Architecture Framework *“presents the reference model for developing business applications that are modular, composable, easily adapted and ready for change.”* [2].

III. FUNDAMENTALS OF NS THEORY

Software should be able to evolve as business requirements change over time. In NS theory [6], the lack of Combinatorial Effects measures evolvability. When the impact of a change depends not only on the type of the change but also on the size of the system it affects, we talk about a Combinatorial Effect. The NS theory assumes that software undergoes unlimited changes over time, so Combinatorial Effects harm software evolvability. Indeed, modifications to a system depend on the size of the growing system. In that case, these changes become more challenging to handle (i.e., requiring more work and lowering the system’s evolvability).

NS theory is built on classic system engineering and statistical entropy principles. In classic system engineering, a system is stable if it has bounded input, which leads to bounded output (BIBO). NS theory applies this idea to software design, as a limited change in functionality should cause a limited change in the software. In classic system engineering, stability is measured at infinity. NS theory considers infinitely large systems that will go through infinitely many changes. A system is stable for NS if it does not have Combinatorial Effects, meaning that the effect of change only depends on the kind of change and not on the system size.

NS theory suggests four theorems and five extendable elements as the basis for creating evolvable software through pattern expansion of the elements. The theorems are proven formally, giving a set of required conditions to follow strictly to avoid Combinatorial Effects. The NS theorems have been applied in NS elements. These elements offer a set of predefined higher-level structures, patterns, or “building blocks” that provide a clear blueprint for implementing the core functionalities of realistic information systems, following the four theorems.

A. NS Theorems

NS theory [6] is based on four theorems that dictate the necessary conditions for software to be free of Combinatorial Effects.

- Separation of Concerns
- Data Version Transparency
- Action Version Transparency
- Separation of States

Violating any of these four theorems will lead to Combinatorial Effects and, thus, non-evolvable software under change.

B. NS Elements

Consistently adhering to the four NS theorems is challenging for developers. First, following the NS theorems leads to a fine-grained software structure, which introduces some development overhead and may slow the development process. Second, the rules must be followed constantly and robotically, as a violation will introduce Combinatorial Effects. Humans are not well suited for this kind of work. Third, the accidental introduction of Combinatorial Effects results in an exponential increase in rework.

Five expandable elements [7] [8] were proposed, which make the realization of NS applications more feasible. These elements are carefully engineered patterns that comply with the four NS theorems and that can be used as essential building blocks for various applications: data element, action element, workflow element, connector element, and trigger element.

- **Data Element**: the structured composition of software constructs to encapsulate a data construct into an isolated

module (including get- and set methods, persistency, exhibiting version transparency, etc.).

- **Action Elements**: the structured composition of software constructs to encapsulate an action construct into an isolated module.
- **Workflow Element**: the structured composition of software constructs describing the sequence in which action elements should be performed to fulfil a flow into an isolated module.
- **Connector Element**: the structured composition of software constructs into an isolated module, allowing external systems to interact with the NS system without calling components statelessly.
- **Trigger Element**: the structured composition of software constructs into an isolated module that controls the system states and checks whether any action element should be triggered accordingly.

The element provides core functionalities (data, actions, etc.) and addresses the Cross-Cutting Concerns that each of these core functionalities requires to function correctly. Cross-cutting concerns cut through every element, requiring careful implementation to avoid introducing Combinatorial Effects.

C. Element Expansion

An application comprises data, action, workflow, connector, and trigger elements that define its requirements. The NS expander is a technology that will generate code instances of high-level patterns for the specific application. The expanded code will provide generic functionalities specified in the application definition and will be a fine-grained modular structure that follows the NS theorems (see Figure 2).

The application's business logic is now manually programmed inside the expanded modules at pre-defined locations. The result is an application that implements a certain required business logic and has a fine-grained modular structure. As the code's generated structure is NS compliant, we know that the code is evolvable for all anticipated change drivers corresponding to the underlying NS elements. The only location where Combinatorial Effects can be introduced is in the customized code.

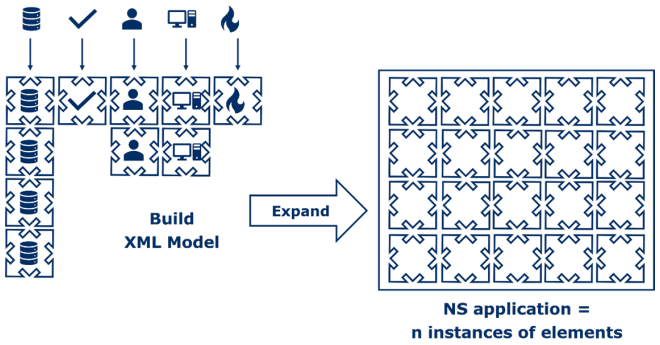


Fig. 2. Requirements expressed in an XML description file, used as input for element expansion.

D. Harvesting and Software Rejuvenation

The expanded code has some pre-defined places where changes can be made. To keep these changes from being lost when the application is expanded again, the expander can gather them and re-inject them when re-expanded. Gathering and putting back the changes is called harvesting and injection.

The application can be re-expanded for various reasons. For example, the code templates of the elements can be improved (fix bugs, make faster, etc.), new Cross-Cutting Concerns (add a new logging feature) can be included, or a technology change (use a new persistence framework) can be supported.

Software rejuvenation aims to routinely carry out the harvesting and injection process to ensure that the constant enhancements to the element code templates are incorporated into the application.

Code expansion produces more than 80% of the code of the application. The expanded code can be called boiler-plate-code, but it is more complex than what is usually meant by that term because it deals with Cross-Cutting Concerns. Manually producing this code takes a lot of time. Using NS expansion, this time can now be spent on constantly improving the code templates, developing new templates that make the elements compatible with the latest technologies, and meticulously coding the business logic. The changes in the elements can be applied to all expanded applications, giving the concept of code reuse a new meaning. All developers can use a modification on a code template by one developer on all their applications with minimal impact, thanks to the rejuvenation process (see Figure 3).

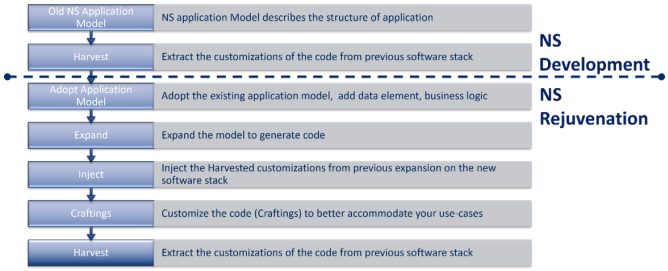


Fig. 3. NS development and rejuvenation.

IV. Related Work

While searching for relevant literature, we found some papers discussing Composable Architecture. The most relevant publications are, on the one hand, a paper about a possible methodology and demonstration by use case for implementing Composable Architecture, and on the other hand, the book of AW Sheers, "The Composable Enterprise" [9]. While Scheer [9] tried to rearchitect a complete organization, Ivas [10] provides a methodology to introduce Composable Architecture and demonstrates it with a use case. The paper also provides a good literature review where the author notices that only a few academic papers discuss Composable architecture (search on

Google Scholar, Web of Science, and Resource Gate). At the same time, thousands of papers are found in the professional literature (via Google), pointing out the need for academic attention to the subject. The paper proposes a methodology consisting of 6 steps (from [10]:

- Understand business drivers and objectives. The first step is to understand the business background of the initiatives, i.e., the rationale, objective, and scope from the business point of view.
- Understand the holistic scope of the initiative. The second step is about understanding which value stream steps and business capabilities from the Holistic value delivery are being affected by the initiative and how.
- Understand the current situation. Understand and sketch the scope of the current solution (architecture).
- Understand the situation and needs at the enterprise level. Identify if any components can be reused or optimised by this solution at the enterprise level or if there are other future initiatives with the same need.
- Design as API -first Headless PBC (preferably according to MACH [11]). If you need to implement a new service, you should preferably design it according to MACH principles. Otherwise, deliver business change by creating new or optimising existing monolith modules by API-first and Headless MACH principles.
- Implement business-IT aligned PBC solution and consolidate. Implement the agreed business-IT solution and consolidate any old solutions into the new solution that implements the same functionality (business capability).

For an Enterprise Architect, those steps are logical and provide excellent guidance. We argue that, next to the need for academic attention to applying Composable Architecture for (re)introducing functional re-uses, there is a need as well for academic attention to properly operationalizing/implementing Composable Architecture to make the dream of McIllroy a reality and not a nightmare.

V. Operationalization of Gartner's Reference Architecture for Composable Business Technology

In this section, we will take the components of the Composable Architecture Framework (see Figure 1) and try to guide how to operationalize them. We start by looking closer at business capabilities and how they should help align business and IT. We continue by looking at the modularity of PBCs and the different types of PBCs defined by the Composable Architecture Framework. We end this section by looking at PBCs and Cross-Cutting Concerns (CCC) and by taking a closer look at the requirements for a PBC platform.

A. Defining Business Capabilities

The concept of business capabilities originates in the resource-based view [12] of companies, where it is considered vital to identify resources and capabilities that provide a competitive advantage [13]. This evolved into the idea that companies need to know "what" kind of activities they are undertaking and gave rise to using the term Business Capabilities. Although decades have passed since the first mention of capabilities, there is no agreed-upon definition of business capabilities and how they should be defined, named and properly used. We refer to [13] for a comprehensive literature overview.

The cornerstone of Gartner's Composable Architecture Framework is PBCs. However, as argued above, the definition of business capabilities can be problematic. If the definitions are unclear, then the implementation into PBC is suspect. For some industries, there are business capabilities frameworks such as BIAN [14] for the banking industry or NBility for energy transmission and distribution in the Netherlands [15]. Still, a lot of sectors have no such frameworks openly available.

As such, a necessary condition for using the Composable Architecture Framework, being well-defined business capabilities, is already a tough nut to crack, and scientific guidance on how to do it is lacking.

B. Business IT Alignment using PBCs

Gartner considered the business capabilities to be well-defined, shared between business and IT, and as an accurate alignment tool between the two. They refer to "Fusion Teams" as the secret formula to create this shared understanding. Fusion teams are teams in which business and IT people are present. Close collaboration between the two will result in shared understanding and better solutions. This idea is also present in Agile Frameworks such as SAFe [16], where agile teams are considered multi-functional (mix of business and IT), and having people who know the job best close together, their emerging designs will be better than those imposed by intentional architecture. Or stated otherwise, the PBCs are to be built bottom-up and not defined top-down.

From NS, we know the importance of having an anthropomorphic design. This means that the naming of modules should represent something that exists in the real world, as this increases understanding of the module's function. The same holds for business capabilities. If they are insufficiently fine-grained and abstract, they no longer represent business reality, but if they are too detailed, the number of business capabilities is considered too large and thus complex. Whether bottom-up or top-down is the right approach is beyond the scope of this paper. One often notices a combination of both; they meet in the middle somewhere.

C. PBCs and Modularity

Gartner's PBCs are modular, where modularity is defined as "partitioned" into a cohesive set of components [2]. Garter further adds that *"The granularity of PBCs, as with all modular systems, is a common design challenge. Modular components that are too large may be easier to manage, but they are harder to change are use in new compositions. Components that are too small may be easier to assemble but harder to isolate, identify, find or change."* [2]. This statement is vague. What are the objective criteria for too large and

too small? Secondly, as neither is considered a good modular design, what are the requirements for a sound module size?

Normalized Systems theory explicitly defines the optimal module size to facilitate anticipated changes. A module must be split into smaller modules until each complies with the four NS theorems: SoC, AvT, DvT, and SoS. When all the theorems are met, the optimal size is reached. As these are the necessary conditions for system stability under change, violating them will introduce Combinatorial Effects (CE).

Separation of Concern (SoC) teaches that each PBC should encapsulate a separate change driver. PBCs must be defined as fine-grained and very specific. For example, a possible PBC is Asset Monitoring. The asset could be an IT system (servers, databases) or an OT system (Operational Technology as SCADAs, Turbines, etc.). Although both are assets, secure monitoring for both requires different implementations. This would mean that if we were only to use one PBC for this, we would need two different versions of this PBC: one for IT and one for OT. It suffices to extend this way of thinking and conclude that insufficiently fine-grained PBCs would lead to multiple versions of the PBC and break the anthropomorphic relation between what something is (a PBC) and how something is done. Such an anthropomorphic relation is a required condition for a PBC platform as PBCs must be easily discoverable (see Section II)

Action version Transparency (AvT) enforces interface stability when the implementation changes. The Composable Architecture Framework does not explicitly mention interface stability when the PBC implementation changes. However, it does say the need for technologies that will result in loosely coupled systems, such as providing interfaces via APIs and event buses and using technologies promoted by the MACH Alliance [11]. We will return to this point when discussing the PBC platform. However, AvT should not be limited to technical protocols and implementations. PBC definitions should also pay attention to semantic issues regarding version transparency. In case a business capability introduces a new feature, a default behavior needs to be defined in case the new feature is not specified.

Data version Transparency (DvT) ensures that evolution in data attributes does not impact processing functions that do not use these new attributes. The Composable Architecture Framework does not mention this concept; it is left to the data fabric. This paper will not discuss the vagueness or evolvability of concepts such as data fabric.

Separation of State (SoS) required all modules to keep state and make their state externally visible. The Composable Architecture Framework does not mention the need for statekeeping of the PBCs. Like with AvT, there is the mention of MACH technologies that promote using process Choreography over process orchestration for evolvability purposes. Independently of whether this is valid, choreography (as orchestration) requires state-keeping and exposition if you want to trace process issues back to the failure location. Similar to AvT, statekeeping should also be looked at semantically, i.e., what is the actual business state at specific points in the process flows.

D. PBC Types

Garter introduces PBC Types, such as application PBC, data entry, analytics PBC, Digital Twin and Process. What is missing is the relation between a PBC and a PBC Type. Is a PBC comprised of multiple PBC types, or do they have a one-to-one relationship? The latter would be strange as to do something, a ”what” or a capability, you require things to make it happen, ”hows”, and the PBC Types are pointing more into the direction of a ”how” than in the direction of a ”what”. The PBC Types have similarities with NS elements.

- Application PBC Type (action) versus the Task Element
- Data Entity PBC Type (reference) versus Data Element
- Process PBC Type (flow) vs Flow Element

For NS, an Analytics PBC would be an Application PBC, where the action/task is to perform some analytic operation. A similar reasoning applies to the Digital Twin (thing) PBC, where this would combine data and task elements in NS. We notice Garter’s difficulty in addressing concepts at their suitable granularity and abstraction level.

E. PBCs and Cross-Cutting Concerns

Normalized Systems recognize that cross-cutting concerns must be treated as any other change driver and require splitting into different modules and proper encapsulation. PBC focus on business, not technology. The technologies used to implement, run, and deploy the PBC are abstracted, and the only type of guidance regarding technology is to use MARCH-compliant technologies. As stated in the previous subsection, we notice an analogy between PBC Types and NS Elements. NS Elements are there to allow proper encapsulation and separation of cross-cutting concerns. One would expect something similar in PBCs, but this is not the case. This represents a lack of design criteria for the PBC and could result in proper SoC regarding capabilities/functionalities but zero evolvability for the cross-cutting concern and associated technologies. As new technological implementations change faster and faster, ignoring this change driver will introduce CEs. Gartner implies leaving this up to the PBC Platform to take care of.

F. PBC Platform

Much of the PBC implementation magic is left over to the PBC Platform. It must facilitate the discovery of the available PBC, the composition of PBCs and the deployment of PBCs. We already argued that proper discovery requires anthropomorphism of the PBC namespace and associated granularity to avoid PBC versioning without meaning. In the previous subsection, we argued that Gartner seems to push the treatment of the cross-cutting concerns toward the PBC Platform. Instead of addressing cross-cutting concerns at the PBC level, it would need to be done at the PBC Platform level. This would be a possibility, were it not that as a selection criteria for a PBC platform, it is currently not mentioned in the Composable Architecture Framework.

Another crucial aspect of the PBC platform is the design, deployment and running of the PBCs. Gartner sees that a change in a PBC Type would yield an update of all composed

applications that use such PBC Type. In NS, the updates of templates making up the elements are handled through expansion and rejuvenation. The new version of the element templates triggers a rejuvenation cycle, updating all instances where this element template is used. The final step is to deploy the application.

Gartner simplifies this concept with an example of Planning PBC used in two compositions [5]. The update of that Planning PBC would trigger the update of both compositions. The question is, what is being updated? Is it a PBC Type that underlies the Planning PBC? Is it the Planning PBC template or the code making up the PBC? Does this update result in one deployment of the Planning PBC shared by two compositions, or are there two deployments of the Planning PBC? In the former case, it would mean that the Planning PBC has zero customizations (even in terms of low and no code). Otherwise, it cannot be used in two different composite applications without putting extra differentiation logic into the Planning PBC or the composite applications. In the latter case, it would mean that running instances of Planning PBCs are not identical, and the difference must be managed on the Platform, resulting in different PBC versions. We already discussed the issues related to that topic.

The problem is that clear guidance is missing on how the PBC platform should handle this. The PBC Platform must also conform to the four NS principles, or it will not allow the evolvability Gartner portrayed in its Composable Architecture Framework.

VI. Validation By Focus Group

The previous section looked at the components of the Composable Architecture framework and tried to point out possible operationalization issues employing NS and business capabilities theories. To avoid our findings being considered too opinionated, we conducted a small experiment with students of the Antwerp Management School who had just been exposed to NS. The idea was to investigate how they look at a concept such as Composable Architecture once they know about NS.

Between September 2024 and November 2024, the Master in Enterprise IT Architecture (MEITA) students of the Antwerp Management School were exposed to Normalized Systems for 16 hours. The MEITA is an executive master, and students in the program already have many years of practical experience in IT Architecture. At the end of this period, they were given the Gartner paper about the Composable Architecture Framework and asked to read this document using their newly acquired knowledge of NS. All students got the reading material beforehand, were asked to read it, and were asked to discuss it in groups for thirty minutes and provide a summary of their findings in 5 minutes. In total, 18 students participated, split into four groups.

The first group had difficulty understanding the meaning of Packaged Business Capabilities (PBCs). They realised that the approach can only succeed if business and IT people define the PBCs. They expected more guidance on how to do this and foresaw evolvability issues, as no mechanisms were foreseen to adequately address SoC beyond business changes (what about technological changes?).

The second group tried to list the pros and cons of the framework. They found the usage of no/low-code with a marketplace of PBCs to be powerful. They considered the recommended usage of headless promoting technologies to be a good way to decouple UI and logic, adhering to SoC at least for those two concerns. Group one also struggled with understanding PBCs and claimed they had never observed it in their practices. They see the framework's application more in classifying your existing applications according to business capabilities and then use the Composable Architecture Integration platform to recombine existing applications.

The third group also struggled to understand PBCs. They compared the main characteristics of PBCs with the NS theorems. Modularity resonates with NS, but the framework has issues describing the level of granularity. They see SoC applied to some extent but insufficient to be NS compliant. They saw that the second characteristic, autonomy, would require SoS, but this is not mentioned. They make a similar remark about PBC orchestration. They conclude that the fourth characteristic, discoverability, misses the need for AvT and DvT.

The fourth group noticed that while NS is about change over time and minimizing ripple effects, PBC is about building fast. The characteristics of PBC are such that nobody in their right mind would not want them, but the framework insufficiently explained how to do it. On the other hand, NS tells you how to do it, and doing it the NS way requires significant effort. The group also identifies the PBC granularity uncertainty as a source of future issues. For instance, the sales business capability, packaged in a PBC, may or may not answer to the different sales business capabilities needs in large and complex organizations, opening the door for violation of the four NS principles quicker than expected.

The different groups struggle with similar issues. Once exposed to NS, one can ask more profound questions about how implementation should occur and whether a proposed solution has the characteristics it claims to have. It is important to note that this does not necessarily mean that NS is the optimal solution to operationalize the concept of PBC, as only a single methodology was evaluated in the focus group. Nevertheless, it does seem to validate that there is a need for a strategy to operationalize a PBC architecture through more concrete guidance, such as provided to a certain extent by NS.

VII. Conclusion

Application re-use is not just a long-forgotten dream of McIllroy, but a focus on many companies. The ability to reuse and recombine applications to support the changing business conditions of an expectation many CEOs have toward their CIOs. In recent years, the focus on functional reuse has been pushed aside by technology-focused integration patterns. Gartner puts functional re-use back on the map with its Composable Architecture Framework, where PBCs are the

essential building blocks of application landscapes. By using NS as an instrument of design and evaluation method for evolvable systems, we pointed out operationalization issues one might face when trying to implement the Composable Architecture Framework, or one might use it to critically evaluate providers of solutions based on it. A focus group was used to validate and balance our findings.

ACKNOWLEDGMENT

The authors thank Rudy Claes of Innocom for introducing them to Gartner's Composable Architecture Framework and conducting a brainstorming session about the framework and its evolvability. We also thank the Master in Enterprise IT Architecture (MEITA) students at Antwerp Management School (AMS) for their contribution as focus group.

REFERENCES

[1] G. Coulouris, J. Dollimore, and T. Kindberg, "Distributed Systems: Concepts and Design Edition 3," ISBN:978-0-201-61918-8, 2001.

[2] J. Sun and Y. Natis. "Use Gartner's Reference Model to Deliver Intelligent Composable Business Applications," Gartner, ID G00720701, 2020 - refresh 2022.

[3] H. Mannaert, P. De Bruyn, and J. Verelst, "On the interconnection of cross-cutting concerns within hierarchical modular architectures," IEEE Transactions on Engineering Management, Vol. 69, pp. 3276-3291 2020.

[4] P. Huysmans, G. Oorts, P. De Bruyn, H. Mannaert, and J. Verelst, "Positioning the normalized systems theory in a design theory framework," Lecture notes in business information processing, ISSN 1865-1348-142, pp. 43-63, 2013.

[5] Y. Natis and G. Alvarez, "How to Implement Composable Technology with PBCs," Gartner, ID G00751018, 2021.

[6] H. Mannaert, J. Verelst, and P. De Bruyn, "Normalized Systems Theory: From Foundations for Evolvable Software Toward a General Theory for Evolvable Design," ISBN 978-90-77160-09-1, Koppa, 2016.

[7] H. Mannaert, J. Verelst, and K. Ven, "The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability," Science of Computer Programming, Volume 76, Issue 12, pp. 1210-1222, 2011.

[8] P.Huysmans, J. Verelst, H. Mannaert, and A. Oost, "Integrating information systems using normalized systems theory: four case studies," In IEEE 17th Conference on Business Informatics, Volume 1, pp. 173-180, 2015.

[9] A.W. Scheer, "The Composable Enterprise: Agile, Flexible, Innovative: A Gamechanger for Organisations, Digitisation and Business Software," ISBN:978-3-658-42482-4, Springer, 2024.

[10] I. Ivas, "Implementation of Composable Enterprise in an Evolutionary Way through Holistic Business-IT Delivery of Business Initiatives," In Proceedings of the 26th International Conference on Enterprise Information Systems, Volume 1, ISBN: 978-989-758-692-7, pp. 397-408, 2024.

[11] MACH Alliance, [Online], Available: https://machalliance.org, [retrieved: March, 2025].

[12] J. Barney, "Firm resources and sustained competitive advantage," In Journal of management, Volume 17, Issue 1, pp 99-120, 1991.

[13] T. Offerman, C.J. Stettina, and A. Plaat, "Business capabilities: A systematic literature review and a research agenda," In International Conference on Engineering, Technology and Innovation (ICE/ITMC), pp. 383-393, 2017.

[14] BIAN, [Online], Available: https://bian.org, [retrieved: March, 2025].

[15] NBility, [Online], Available: https://www.edsn.nl/nbility-model, [retrieved: March, 2025].

[16] SAFe Framework, [Online], Available: www.scaledagileframework.com, [retrieved: March, 2025].