International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Architecture Patterns Clustering using a Machine Learning Approach

Omar AlHuniti[1], Khawla Al-Tarawneh[2], Esra Alzaghoul[3], Fawaz Ahmad Alzaghoul[4]

Submitted: 05/02/2024 **Revised**: 13/03/2024 **Accepted**: 19/03/2024

Abstract: Architecture patterns are frequently employed in software development to address prevalent design challenges. The identification and classification of architecture patterns have become crucial in optimizing the design process due to the increasing complexity of software systems. Clustering has emerged as a widely adopted technique to categorize comparable entities. Recently, machine learning algorithms have been employed to automate and enhance the precision of clustering.

This study proposed using k-means clustering to group the architectural patterns like repository, client-server, broker, microkernel, publisher-subscriber, model view controller, REST, and space-based patterns together.

That was done on one of the benchmark dataset (Architectural Patterns dataset) by using different Ks to perform the clustering, demonstrating the connections between architecture patterns. Then extract the related patterns and propose valid splitting for some patterns.

Keywords: Architecture patterns, Clustering, Image Processing, Machine Learning.

I. Introduction:

The domain of software architecture has grown significantly in recent years as a result of the introduction of new software systems and the rising complexity of old ones. One key problem in software architecture design is discovering and selecting appropriate architecture patterns that may be used across different projects. These patterns provide a collection of tried-and-true design solutions that may be applied to specific issue areas, ultimately improving the quality of software systems.

A set of standards that influence the construction of a software system is referred to as an architectural pattern. Functional requirements, restrictions, and quality attribute criteria are all part of these specifications. Architectural patterns are reusable solutions that are labeled and can be used to address frequently occurring difficulties in software architecture design. Software architects have long used architectural pattern catalogs such as Pattern-Oriented Software Architecture and Patterns of Enterprise Application Architecture.[4]

Clustering is a proficient data analysis approach that groups together comparable objects based on their properties or features. Machine learning, data mining, pattern recognition, and picture segmentation are all disciplines where clustering is used. The k-means algorithm is a well-known clustering method that divides data into k groups, where k is a user-specified value. The algorithm assigns each data point to the nearest cluster center iteratively and then updates the cluster centers depending on the new assignments. When the assignments no longer change or when a predetermined convergence threshold is reached, the algorithm has converged.[5]

Selecting an appropriate clustering algorithm is critical to any machine learning-based approach to architecture pattern analysis. This paper uses the k-means clustering algorithm due to its simplicity, efficiency, and effectiveness in clustering large datasets. Using benchmark datasets, Architectural Patterns Dataset.

The paper is organized as follows: In section two, background and literature review, a group of research papers related to the research are presented. In section three, we describe the methodology of our proposed approach. In section four, results and discussion are conducted. In section five, the conclusion and future work are displayed.

II. Background And literature review:

Architecture patterns are design frameworks or templates often used in software architecture to provide a proven solution to a recurring design problem. These patterns capture the essential features of a system's structure and behavior, allowing architects to create systems that are scalable, maintainable, and adaptable to changing requirements.

Many studies were done on architectural patterns and their features, pointing out their advantages and

1Amr9220474@ju.edu.jo
King Abdullah II School of Information Technology
2Kol9220471@ju.edu.jo
King Abdullah II School of Information Technology
3Esra@ju.edu.jo
King Abdullah II School of Information Technology
4Fawaz.alzaghoul@gmail.com
Department of Software engineering, College of Sciences and Information Technology, Jadara University, Irbid Jordan

disadvantages. One of the simple patterns was Visualizing Architecture (VIZ), The VIZ pattern is a set of techniques and concepts to generate graphical depictions of software architecture. The primary objective of architectural visualization is to aid stakeholders, such as developers, architects, and business users, in comprehending the system's structure and behavior while facilitating communication and decision-making. In their recent publication, Gebremeskel et al. (2022) proposed that the architecture of data mining models can be optimized to facilitate the effective visualization of information extraction in the context of patient safety care. They proposed an approach that provides timely information to facilitate informed medical decision-making regarding patient safety care. The importance of visualization based on modeling is emphasized. The term "model representation" pertains to the visual manifestation of a model, which showcases the constituent elements that necessitate comprehension as they interconnect and amalgamate with one another. The interactivity factor pertains to the ability to observe the model being demonstrated in real time and integrate it as the user interacts with it as needed. Integration refers to the ability to demonstrate connections between patient records and other perspectives within the data based on which it is founded.

While the broker pattern is the primary mechanism that facilitates communication between multiple components by acting as an intermediary, it enables the components to exchange information without knowing each other's implementation details. The technique provides a scalable and adaptable means of communication between remote system components. Facilitating clear communication among system components results in streamlined system maintenance and enhanced adaptability over an extended period. A message broker is a commonly used implementation of the broker design pattern, which serves as a centralized hub for transmitting and receiving messages among various components or systems in message-oriented middleware systems. The study by Lévy et al. (1998) aimed to examine the mechanisms involved in comparing architectural styles. The objective of comparing styles is to establish a set of standards that can aid in selecting a suitable style, contingent upon the specific demands of the application. Subsequently, it was employed to delineate the attributes of both the Mediator and the Broker. It was subsequently determined that the Broker is qualified to serve as a proficient mediator. Using a broker is prevalent in facilitating communication between two entities, particularly in scenarios with a data dependency. This approach is deemed advantageous in achieving object decoupling.

On the other hand, the client-server pattern is a prominent software engineering architecture pattern utilized in developing distributed systems. This pattern involves a client communicating with one or more servers to execute a task or gain access to a resource. In this particular design, the server receives requests from the client, processes them, and subsequently sends back a response to the client. The server furnishes clients with a collection of amenities and assets, and the client communicates with the server through a precisely defined protocol. Kassab et al. (2018) conducted a survey targeting software professionals to identify architectural patterns. The numerical outcomes from this preliminary survey provide scope for further examination and evaluation. Implementing the peer-to-peer pattern was comparatively complex, and its adoption incurred higher costs, while the client-server architecture was observed to be relatively more uncomplicated.

The system can be compartmentalized into discrete, autonomous units, with a fundamental microkernel module serving as the essential foundation. The microkernel provides module communication, loading and unloading, and system resource management. The residual system functionalities are executed as distinct units that establish communication amongst themselves through the microkernel. Baccelli and colleagues (2013) conducted a study on a Microkernel architecture designed to facilitate the Internet of Things (IoT). Furthermore, utilizing a modular microkernel architecture enhances the system's resilience against errors in individual components.

The Repository pattern decouples the data storage layer from the business and application functionality. The repository serves as an intermediary component that connects the business logic and data storage layer, furnishing a uniform data retrieval and modification interface. The design style in question finds applications in both web applications and database systems. The authors Garcia-Holgado and Garcia (2016) proposed that the assessment of the repository architectural pattern should be conducted in diverse eLearning contexts to authenticate the pattern's definition and facilitate the advancement of this type of technological solution.

The study conducted by Perera and Jayakody (2022) put forth the proposal that a middleware for publisher-subscriber architecture was developed and evaluated for its load and burst performance using two distinct methodologies. Although the publisher-subscriber technology is considered dated, it remains a viable option for developing robust middleware with adaptable functionalities across diverse domains. According to Avgeriou and Zdun's (2005), the peer-to-peer pattern assigns equivalent responsibilities to each component, allowing them to function as both a client and a server. Each constituent element offers its own set of services or data and can access the services provided by other constituent elements. The peer-to-peer network comprises a dynamically evolving collection of constituents. The authors Theorin et al. (2017) proposed

that the event-based service-oriented architecture offers a versatile and scalable solution for managing low-level application control and higher-level information aggregation. Software applications can be developed and evaluated independently, as substitute programs that generate events can readily replace other programs.

Daoudi et al. (2019) evaluated the prevalence of various MVC-based architectural patterns in Android applications. This research proposes several MVC-based patterns in Android applications and seeks to determine their dominant pattern. Their finding indicates that large applications utilize patterns, while small applications tend not to employ any. They suggested that patterns are a viable and efficient approach for large applications. Verborgh and colleagues (2015) The scholar advocated for investigating the role of the Representational State Transfer (REST) design pattern in developing an information framework that can generate and maintain stable identifiers for objects managed by enterprises. They are impeding the accomplishment of the implementation of the Linked Data Principles. Based on the presented data, this usage pattern appears highly effective within institutional settings.

Kaiwartya and colleagues (2016) presented a methodology that utilizes a multi-layer pattern to account for each layer's distinct representations and functionalities. This phrase pertains to utilizing aircraft safety protocols in their management and operation. The efficacy and efficiency of the pattern in intelligent vehicular applications have been established.

The impact of utilizing various styles in business intelligence was examined by Iyer et al. (2019). The utilization of a Presentation Abstraction Controller (PAC) pattern has been found to be a straightforward and efficient approach for strategizing business applications and managing daily sub-tasks in accordance with the growth of the business.

Taibi and colleagues (2018) presented a standardized index that comprehensively evaluates the benefits, drawbacks, and insights gained from various types of case studies pertaining to diverse architectural styles. The microservices architecture patterns encompass various configurations for migration, formatting, storage, and disseminating a prescribed set of principles. Mordinyi et al. (2010) introduced the notion of space-based architecture (SBA), which enables the separation of distributed applications from the underlying middleware architecture. This is achieved by incorporating the features and characteristics of advanced middleware architectural styles into a straightforward API. It retains the benefit of necessitating minimal modifications to the application in the event of a shift in the fundamental middleware architectural paradigm, thereby enabling more effective satisfaction of novel commercial demands.

Many researchers proposed architectural patterns. Aychew,andAlemneh(2022) discussed the significance of adopting appropriate architectural patterns depending on intended system attributes or techniques. According to the authors, choosing a suitable architectural pattern is crucial to achieving system quality traits, including performance, scalability, security, maintainability, and adaptability. The paper then presented an outline of architectural patterns and reusable solutions to common software architecture design difficulties. When choosing an architectural pattern, the writers emphasize the importance of considering system tactics. Tactics are quality aspects significant to system stakeholders, such as performance, security, and availability, because they are the building blocks of architectural patterns, just as atoms are the building blocks of molecules. They also describe a method for selecting architectural patterns based on system tactics. The process comprises identifying system tactics, selecting candidate architectural patterns that are appropriate for the tactics, evaluating the candidate patterns against system requirements, and selecting the optimal pattern for the system. Six distinct experiments were carried out in the proposed study to determine which combination produced a better-performing algorithm. One of these algorithms achieved a 94% accuracy.

Komolov et al. (2022) introduced a machine-learning methodology to forecast architectural design patterns. According to the authors, using design pattern prediction can aid software architects in making well-informed decisions regarding the design of software systems. The study presented a methodology based on machine learning to identify architectural design patterns using source code metrics. The dataset was acquired from GitHub using the GitHub API and subsequently disseminated to the academic community. The dataset comprises 5973 samples, of which 1195 were allocated for testing purposes, and 4778 were designated for training. The investigators utilized nine conventional machine learning techniques to identify the MVVM and MVP architectures within every dataset sample. The ensemble learning techniques of cat boost, support vector machines (SVM), and Neural Networks demonstrated exceptional F1 score, recall, precision, and accuracy. The machine learning model that exhibited the highest level of performance achieved an 83% precision, recall, accuracy, and F1 score.

Cluster architectural patterns can be used to create scalable and fault-tolerant systems. They provide better resource usage and system performance while delivering high availability and fault tolerance Daoudi et al. (2019) provided a method for determining which MVC-based patterns dominate an Android application. According to the study, MVC is the most commonly used pattern in Android apps, with 57% of Android apps, and it is increasing in popularity in the Android community.

MVP is less popular, and many projects still need an MVC-based paradigm. The study also discovered no association between whether or not a pattern is used and an app's category. Furthermore, most small-sized apps do not employ any pattern. These findings emphasize the significance of comprehending the existence and application of architectural patterns in Android app development.

Moreover, Velasco-Elizondo et al. (2016) presented a novel approach for analyzing architectural patterns using knowledge representation and information extraction techniques. The authors propose a framework that can automatically extract and represent relevant information from architecture pattern documents and use this information to perform analysis tasks such as pattern classification and identification of pattern relationships. The framework uses a knowledge representation model that captures architectural patterns' essential features and relationships. The authors evaluate their approach using a set of real-world architecture pattern documents and demonstrate that their framework can effectively extract and represent the necessary information for pattern analysis. This work contributes to the field of software architecture by providing a systematic approach for analyzing architecture patterns that can be used to support the design and evaluation of software systems. Furthermore, explain a methodology that employs knowledge representation and information extraction methodologies to scrutinize architectural pattern descriptions about particular quality attributes. An automated approach can be facilitated by utilizing a computable model as a prototype tool. The study centered on the performance quality attribute and utilized experimentation on a pattern corpus involving 45 architects with differing experience levels. The findings indicated that the suggested technique enhances recall and diminishes analysis time compared to manual analysis.

III. Methodology:

1- Dataset:

Our study was done on one of the benchmark datasets, Architectural Patterns Dataset. It contains 2035 images from the main fourteen Architectural Patterns (Broker, ClientServer, EventBusPubSub, Layered, Microkernel, Microservices_ServiceBased, ModelViewController, PeerToPeer, PipeAndFilter, PresentationAbstractionController, PublishSubscriber, Repository, REST, and Spacebased)those images are balanced and distributed around one hundred each architecture. This dataset from the IEEE data port is a valuable resource for anyone interested in software architecture. All are available in high-resolution image forms of the architecture and provide a comprehensive overview of the field.

2- The Methodology:

The first step of this study was to prepare the images by putting all of them in one group and resizing them to be 500x500 pixels on grayscale. This data preprocessing was a crucial step that ensured the coherence of the data and yielded optimal outcomes.

Considering the data type being utilized, specifically images, it was necessary to resize the images to ensure uniform dimensions.

Additionally, normalization techniques were employed to mitigate any potential biases that may have affected the analysis. To reduce the computational burden of the clustering procedure, the images were transformed into grayscale and then converted into a one-dimensional array appropriate for k-means clustering.

Subsequently, our attention was directed toward feature extraction to clustering the images based on their visual resemblance. After preparing the data and extracting the main features, we performed a clustering analysis utilizing the k-means algorithm.

The algorithm employs a partitioning technique to split a given dataset into several clusters (K). Each data point is allocated to the group whose mean is closest to it.

The process was done in different k values, and by experimental, the proper values were the ones that related to the number of original distinct architectural patterns in the dataset. The dataset comprised fourteen distinct architectural patterns, so the final selected values of K in this study were seven, fourteen, and twenty-one, corresponding to the anticipated number of unique clusters.

After computing the proximity among the data points using the Euclidean distance, we cluster them using k-means clustering. Our choice of K as fourteen was based on our dataset comprising fourteen distinct architectural patterns. Each cluster represented a unique pattern, and we used this information to gain insights into the underlying structure of our data. By analyzing each cluster's characteristics, we could identify commonalities and differences between the architectural patterns. This allowed us to conclude the design principles that governed these patterns and how they could be improved or optimized. Using k-means clustering and Euclidean distance proved a practical approach for analyzing complex datasets with multiple variables and identifying meaningful patterns within them.

After conducting the k-means clustering, we proceeded to the validation stage to evaluate the clustering algorithm's efficacy. Multiple validation metrics were employed, among which was the silhouette coefficient. This metric quantifies the degree of resemblance

between an entity and its corresponding cluster compared to other clusters. Furthermore, the Rand Index was employed as a metric considering the proportion of correct determinations. The clustering algorithm used in this study was evaluated using multiple validation metrics, including the silhouette coefficient and the Rand Index. The silhouette coefficient measures how well an entity fits into its assigned cluster compared to others, providing insight into the quality of the clustering results. The Rand Index, on the other hand, assesses the accuracy of the clustering by calculating the proportion of correct determinations. By utilizing these metrics, we evaluated and optimized our clustering algorithm for improved accuracy and efficiency. Overall, these validation metrics proved to be valuable tools in assessing the effectiveness of our clustering approach.

Finally, we proceeded to the analysis and interpretation of the findings. The analysis yielded clusters, which were subsequently scrutinized. Each image within a given cluster was thoroughly examined to identify shared characteristics. In addition, the outcomes of the clustering process were cross-checked with the initial image labels, thereby facilitating the evaluation of the clustering's precision. Notably, clustering is an investigative methodology; therefore, there is a degree of subjectivity in interpreting outcomes. Consequently, our objective was to engage in an iterative approach, whereby we would make essential adjustments to the parameters and verify the outcomes with professionals in the relevant field whenever feasible.

IV. Result and discussion:

This research used Google Colaboratory (Colab), an online interactive Python development and execution environment, to implement and execute our computational models.

This cloud-based platform offered a flexible workspace with access to high-performance computing resources, which expedited the execution of complex algorithms integral to our study. The adoption of Python language in this process was instrumental, given its comprehensive libraries and tools that facilitated efficient data manipulation, analysis, and visualization. The choice of these technologies significantly contributed to the robustness of our research result.

In this study, we used K-means clustering to collect images, revealing interesting groupings and illuminating underlying architectural patterns. The clusters that appeared at different values of k had a definite distribution of images throughout them. The fundamental qualities were crucial in constructing each group were revealed by carefully analyzing images from each cluster, especially those related to architectural traits. Notably, commonalities and recurrent patterns were found in the photographs of each cluster, confirming the algorithm's ability to recognize and classify images according to their inherent architectural similarity.

When using seven as the K value cluster, all architectural patterns will be divided into seven clusters, as shown in the sample of groups in figure one.

That shows how you could group these patterns based on some common features and the most common patterns were as the following lists of groups:

1- (EventBusPubSub, REST, PublishSubscriber , PeerToPeer)
2- (Microservices_ServiceBased,spacebased)
3- (Layered, ModelViewController, PresentationAbstractionController)
4- (repository)
5- (ClientServer)
6- (Microkernel)
7- (PipeAndFilter).

This clustering depends on the images but agrees with the functionality, architecture style, and communication model.

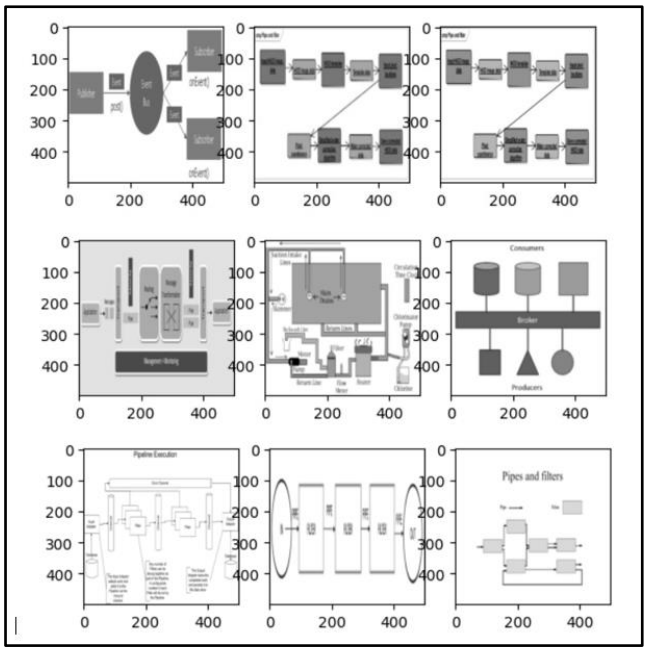

Figure 1: Three different clusters in k value equal seven

While using fourteen as a K value cluster, all architectural patterns in it return the original fourteen clusters, which agrees with the collected dataset's distribution as the sample of groups that appear in Figure two.

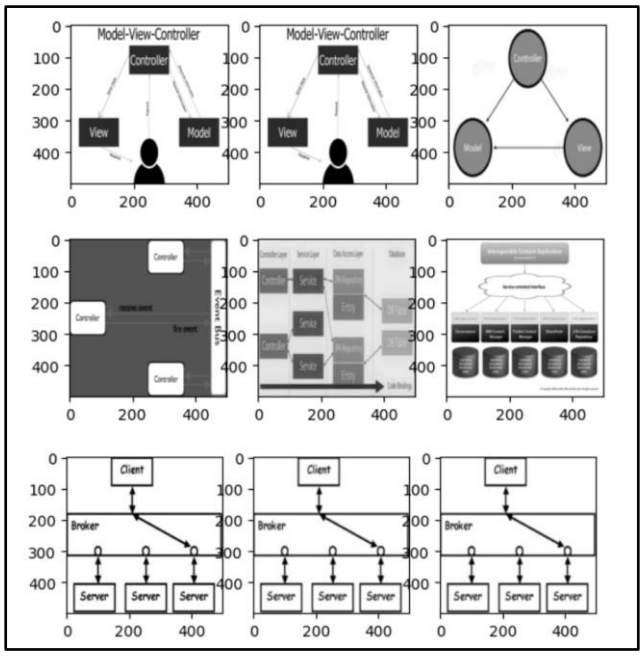

Figure 2:Three different clusters in k value equal fourteen

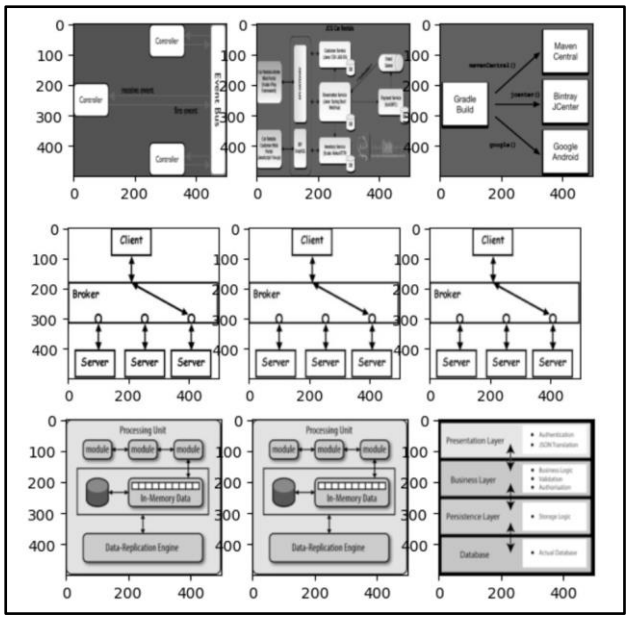

Figure 3: Three different clusters in k value equal twenty-one.

When expanding the number of clusters to twenty architectural patterns, many architectural patterns show the clear dividing into two sub-patterns that are shown clearly on some of them like

Layered architecture patterns give two subgroups. One shows the Three-tier architecture, and the second contains N-tier architecture.

V. Conclusion and Future Works:

This research introduces an approach for k-means clustering aimed at grouping entities with identical architectural features. The similarity between the fourteen distinct architecture patterns was assessed using three distinct k values. The value of k is deemed suitable for the cluster across all of our data points.

The clustering process provides a comprehensive understanding of the underlying patterns, facilitating comparative analysis of various architectural patterns and identifying their distinctive characteristics.

Our recommendation is to thoroughly analyze the pattern cluster and generate a novel hybrid architecture pattern based on the closest group.

References:

[1] Ritu Kapur, Sumit Kalra, Kamlesh Tiwari, Geetika Arora. (2021). Architectural Patterns Dataset. IEEE Dataport. https://dx.doi.org/10.21227/z9k4-8217

[2] Aychew, M., & Alemneh, E. (2022, November). Selection of Architectural Patterns based on Tactics. In 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA) (pp. 13-18). IEEE.

[3] Komolov, S., Dlamini, G., Megha, S., & Mazzara, M. (2022). Towards Predicting Architectural Design Patterns: A Machine Learning Approach. Computers, 11(10), 151.

[4] Velasco-Elizondo, P., Marín-Piña, R., Vazquez-Reyes, S., Mora-Soto, A., & Mejia, J. (2016). Knowledge representation and information extraction for analysing architectural patterns. *Science of Computer Programming*, *121*, 176-189.

[5] Daoudi, A., ElBoussaidi, G., Moha, N., & Kpodjedo, S. (2019, April). An exploratory study of MVC-based architectural patterns in Android apps. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (pp. 1711-1720).

[6] Rokach, L., & Maimon, O. (2005). Clustering methods

[7] Gebremeskel, G. B., Hailu, B., & Biazen, B. (2022). Architecture and optimization of data mining modeling for visualization of knowledge extraction: patient safety care. Journal of King Saud University-Computer and Information Sciences, 34(2), 468-479.

[8] Lévy, N., Losavio, F., & Matteo, A. (1998, November). Comparing architectural styles: broker specializes mediator. In Proceedings of the third international workshop on Software architecture (pp. 93-96).

[9] Kassab, M., Mazzara, M., Lee, J., & Succi, G. (2018). Software architectural patterns in practice: an empirical study. *Innovations in Systems and Software Engineering*, *14*, 263-271.

[10] Baccelli, E., Hahm, O., Günes, M., Wählisch, M., & Schmidt, T. C. (2013, April). RIOT OS: Towards an OS for the Internet of Things. In 2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS) (pp. 79-80). IEEE.

[11] García-Holgado, A., & García-Peñalvo, F. J. (2016). Architectural pattern to improve the definition and implementation of eLearning ecosystems. Science of Computer Programming, 129, 20-34.

[12] Perera, H., & Jayakody, A. (2022, September). Common Object Request Broker-based Publisher-Subscriber Middleware for Internet of Things-Edge Computing. In 2022 International Research Conference on Smart Computing and Systems Engineering (SCSE) (Vol. 5, pp. 68-75). IEEE.

[13] Avgeriou, P., & Zdun, U. (2005). Architectural patterns revisited-a pattern language.

[14] Theorin, A., Bengtsson, K., Provost, J., Lieder, M., Johnsson, C., Lundholm, T., & Lennartson, B. (2017). An event-driven manufacturing information system architecture for Industry 4.0. International journal of production research, 55(5), 1297-1311.

[15] Verborgh, R., Van Hooland, S., Cope, A. S., Chan, S., Mannens, E., & Van de Walle, R. (2015). The fallacy of the multi-API culture: Conceptual and practical benefits of representational state transfer (REST). Journal of Documentation, 71(2), 233-252.

[16] Kaiwartya, O., Abdullah, A. H., Cao, Y., Altameem, A., Prasad, M., Lin, C. T., & Liu, X. (2016). Internet of vehicles: Motivation, layered architecture, network model, challenges, and future aspects. IEEE access, 4, 5356-5373.

[17] Iyer, A., Bali, S., Kumar, I., Churi, P., & Mistry, K. (2019). Presentation Abstraction Control Architecture Pattern in Business Intelligence. In Advances in Computing and Data Sciences: Third International Conference, ICACDS 2019, Ghaziabad, India, April 12–13, 2019, Revised Selected Papers, Part II 3 (pp. 666-679). Springer Singapore.

[18] Taibi, D., Lenarduzzi, V., & Pahl, C. (2018). Architectural patterns for microservices: a systematic mapping study. In CLOSER 2018: Proceedings of the 8th International Conference on Cloud Computing and Services Science; Funchal, Madeira, Portugal, 19-21 March 2018. SciTePress.

[19] Mordinyi, R., Kühn, E., & Schatten, A. (2010, February). Space-based architectures as abstraction layer for distributed business applications. In 2010 International Conference on Complex, Intelligent and Software Intensive Systems (pp. 47-53). IEEE