Optimizing Microservices Architecture Using Multi-Criteria Decision Making (MCDM): A Case Study Of PayPal REST API

I Nyoman Gde Artadana Mahaputra Wardhiana
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
6025231022@student.its.ac.id

Muhammad Zakky Ghufron
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
6025231089@student.its.ac.id

Satria Manggala Jati
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
6025231005@student.its.ac.id

Abdullah Faqih Septiyanto
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
7025221022@student.its.ac.id

Riyanarto Sarno
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
riyanarto@its.ac.id

2024 11th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE) | 979-8-3503-8930-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICITACEE62763.2024.10762818

Abstract— **In the digital transformation era, businesses are increasingly adopting microservices architecture to enhance their software systems' flexibility, scalability, and maintainability. Optimization is needed to achieve these goals. Microservices architecture optimization was applied to a study case of PayPal REST API microservices. There are five architectures include one actual architecture and four developed from the actual architecture. These architectures are based on optimizing criteria such as granularity, complexity, and cohesion. The architecture evaluation uses Service Granularity Metrics (SGM), Number of Operations (NOO), and Lack of Cohesion Metric (LOCM) in various weights. Multi-Objective Optimization based on Ratio Analysis (MOORA) and Complex Proportional Assessment (COPRAS) models are used to determine the best architecture. Almost every architecture alternative suit a specific level of granularity, but alternative 4 has balanced results for each criterion. The evaluations demonstrate that the actual architecture maintains high granularity and showing superior performance in managing operational complexity effectively when assessed against alternative architectures.**

Keywords—MCDM, microservice, architecture, optimization, cohesion, complexity, granularity

I. INTRODUCTION

In the digital transformation era, businesses are embracing microservices architecture as a catalyst for enhancing their software systems' flexibility, scalability, and maintainability [1]. Microservice, a style of architectural design that breaks down a complex system into more minor, more manageable services, is at the forefront of this transformative shift, each service dedicated to specific business functionalities [2]. In the dynamic landscape of software engineering, microservices architecture has emerged as a transformative paradigm for designing and implementing complex distributed systems. Another advantage of microservice applications is their loosely coupled architecture, which enhances fault tolerance [3]. However, optimizing their architecture remains a non-trivial challenge due to the myriad factors influencing their design. While numerous optimization techniques exist, their applicability to microservices, particularly those employed by PayPal, remains underexplored.

As a fundamental aspect of microservices architecture, microservice granularity pertains to the size and scope of individual services within the system. Fine-grained granularity may enhance scalability and agility but could introduce complexity and operational overhead, while coarse-grained services may simplify management but compromise flexibility and scalability [4], [5], [6]. Complexity encompasses the intricacy and interdependencies among microservices. If the correlation between the services increases, the complexity of the architecture also increases [7]. Each of these attributes, granularity, complexity, and cohesion, can be quantified using several metrics, specifically Service Granularity Metrics (SGM), Number of Operations (NOO), and Lack of Cohesion Metric (LOCM) [8], [9]. Each of these attributes, granularity, complexity, and cohesion, can be quantified using several metrics, specifically Service Granularity Metrics (SGM), Number of Operations (NOO), and Lack of Cohesion Metric (LOCM) [10]. Finding the right balance between granularity, complexity, and cohesion in developing microservice architectures is a critical challenge [11]. Pushing granularity too far can result in overly small and fragmented microservices, while too little granularity can lead to overly monolithic microservices. Managing complexity and ensuring cohesion are crucial to building effective and maintainable microservices.

In this research, a Multi-Criteria Decision Making (MCDM) approach has been employed to optimize the attributes of a microservice architecture. MCDM allows developers to consider multiple criteria simultaneously, enabling decision-makers to assign relative weights to each criterion based on their priorities and preferences [12]. The Complex Proportional Assessment (COPRAS) model, a multi-attribute decision-making theory, has comprehensively evaluated complex decision problems by comparing alternatives [13]. The Multi-Objective Optimization by Ratio Analysis (MOORA) model, proposed by Brauers et al. [14], has ranked alternatives based on multiple conflicting objectives, providing insights into trade-offs among competing design choices.

Previous research has explored various methods for decision-making in microservices architecture and multi-criteria decision-making. Goossens et al. [15] designed a decision-making for framework using the Analytic Hierarchy Process (AHP), while Taherdoost et al. [16] reviewed MCDM methods applicable to microservices architecture. Chosy et al. [17] determined the ranking of hospitals using the AHP and MOORA methods, combined ranking between the two methods also confirmed the same ranking order. Ajrina et al. [18] utilized MOORA and COPRAS in a study on sandstone mining. Their research demonstrated the effectiveness of

MCDM methods in guiding decision-making processes. Ryco et al. [19] compared supplier ranking using the MOORA and COPRAS methods, finding no significant difference in final rankings between the methods.

This research aimed to provide a systematic framework for evaluating and optimizing microservices architecture in the context of the PayPal REST API. Through empirical analysis and case studies, the study sought to identify optimal architectural configurations that maximize performance, scalability, and maintainability. Through this research, advancements in microservices engineering are pursued to tackle real-world challenges in software development.

II. Literature Review

A. *The Lack of Cohesion Metric (LCOM)*

In a microservices architecture, LCOM measures how closely activities inside a service are coordinated. It measures how cohesively an application's processes are within a microservice, reflecting its complexity and error-proneness [20]. LCOM is computed using the Eq. 1:

$$LCOM = 1 - \frac{\sum_{i=1}^{n} M\,F}{M \times F} \tag{1}$$

A lower LCOM score indicates better cohesion and coherence across processes, denoting a more efficient and well-organized microservice. On the other hand, a higher LCOM value denotes a more cohesive system, which can result in more complexity and a greater chance of mistakes [10], [21]. The average LCOM (ALCOM) is determined using Eq. 2 to assess the general lack of cohesiveness of the overall microservices in the application.

$$ALCOM = 1 - \frac{\sum_{i=1}^{n} LCOM}{N\,S} \tag{2}$$

Where:

MF	:	Occurrence of a parameter
M	:	Total number of operations
F	:	Count of unique parameters within the microservice
NS	:	Number of microservices in the application
n	:	Number of operations within a microservice

B. *The Service Granularity Metric (SGM)*

This metric evaluates separate services' data and functional elements to provide insights into the microservices application's granularity. Data Granularity of a Service (DGS) and Functional Granularity of a Service (FGS) are the two main components of SGM [22]. DGS determines whether the data granularity is fine-grained or coarse-grained by analysing each operation's input and output data size within a microservice. Eq. 3 below is used to determine the DGS.

$$DGS = \frac{IPR}{\sum_{i=1}^{n} FP} + \frac{OPR}{\sum_{i=1}^{n} FP} \tag{3}$$

Where:

IPR	:	Input parameters per operation
OPR	:	Output parameters per operation
FP	:	Total input parameters in the microservice

Higher DGS values indicate coarse-grained data handling in the microservice, whereas lower values favour fine-grained data handling. CRUD (Create, Read, Update, Delete) functions are given weights according to the degree of data manipulation they involve in FGS, which evaluates the functional granularity of actions within a microservice. Each operation's FGS is determined using Eq. 4:

$$FGS = \frac{OT}{\sum_{i=1}^{n} O} \tag{4}$$

Where:

OT	:	Weight assigned to a specific operation
O	:	Total weight of all operations within the microservice

The overall SGM is then computed as Eq. 5 below:

$$SGM = \sum_{i=1}^{n} DGS \times FGS \tag{5}$$

Providing an indication of the overall granularity of operations within the microservices application. The average SGM (ASGM) across all microservices is computed using Eq. 6 to get the granularity of the entire microservices application [10]:

$$ASGM = 1 - \frac{\sum_{i=1}^{n} SGM}{N\,S} \tag{6}$$

C. *Number of Operations Per Microservice Metric (NOO)*

The number of operations per service is the number of member operations connected to a single microservice. This metric is similar to the WMC metric in that it assesses the number of member methods associated with a given class as a measure of complexity [23]. Below, Eq. 7 is used to calculate this metric.

$$NOO = \sum_{i=1}^{n} M \tag{7}$$

Where:

M	:	Number of operations per service

More operations per service in the context of microservices indicates more application complexity, which could result in more mistakes.

D. *Multi-Objective Optimization based on Ratio Analysis (MOORA)*

The MOORA system is a flexible tool that evaluates each alternative's contribution to a particular objective to all alternatives. The square root of the sum of the squares for each chosen choice must be determined during this process. The System Ratio approach and the Reference Point approach, which both use comparable normalization procedures, are the two parts of the MOORA method [24]. Before optimizing criteria, vector normalization processes are used to turn cost-type criteria into benefits and convert the Decision Matrix into a Normalized Matrix. Every evaluation result in the MOORA approach is normalized using the subsequent [25], [26], [27]:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} (x_{ij})^2}} \tag{8}$$

Where:

r_{ij} : The i-th alternative's normalised performance to the j-th criterion

x_{ij} : Performance ratings

The summation of the instances was performed to enhance each assessment, aiming to maximize the criteria by minimizing them, as described in Eq. 9:

$$Q_i = \sum_{j \in \Omega max} w_j r_{ij} - \sum_{j \in \Omega min} w_j r_{ij} \quad (9)$$

Where:

Q_i : i-th alternative all ranking index

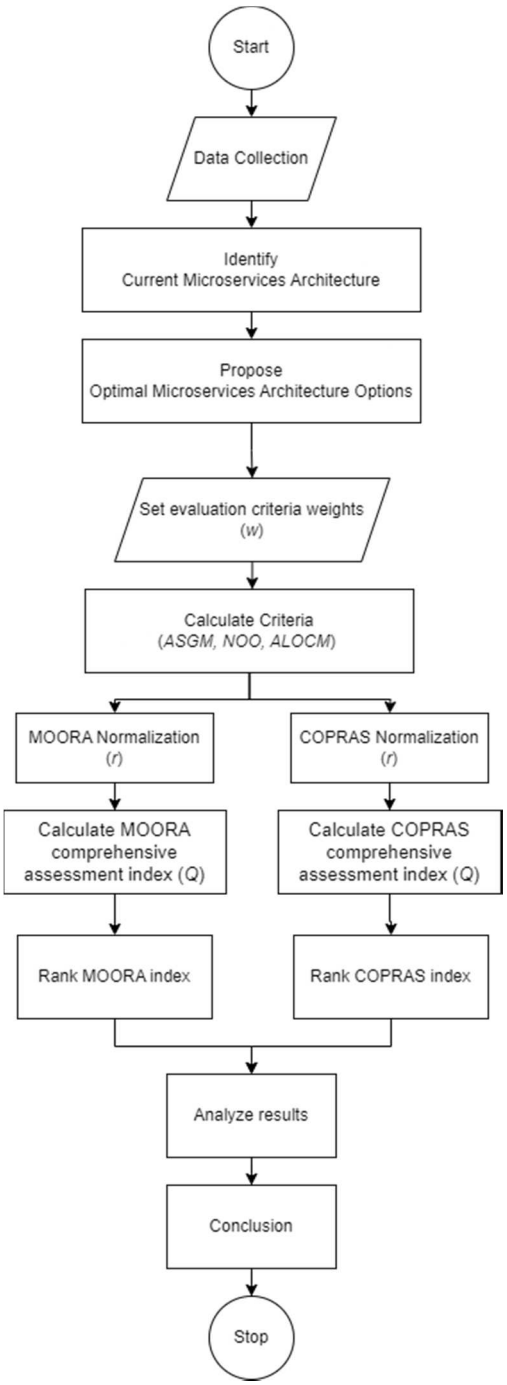

Fig. 1. Proposed Method

Ωmax and Ωmin : Cost and benefit criteria sets

w_j : j-th criterion weight

E. *Complex Proportional Assessment (COPRAS)*

Considering the weights given to each criterion, the COPRAS approach compares alternatives and determines priorities under conflicting criteria. It is predicated on the idea that a dependency's relevance and priority level correspond directly with the alternatives [13]. The following Eq. 10 calculates assessments normalised in the MOORA approach [27], [28]:

$$r_{ij} = \frac{x_{ij}}{(\sum_{i=1}^{n} x_{ij})} \quad (10)$$

The following Eq. 11 calculates the comprehensive assessment index for each alternative.

$$Q_i = S_{+i} + \frac{\sum_{i=1}^{m} S_{-i}}{S_{-i} \sum_{-=1}^{m} \frac{1}{S_{-i}}} \quad (11)$$

Where S_{+i} and S_{-i} refer to the positive score and negative score of the i-th alternative, described in Eq. 12 and Eq. 13:

$$S_{+i} = \sum_{j \in \Omega max} w_j r_{ij} \quad (12)$$

$$S_{-i} = \sum_{j \in \Omega min} w_j r_{ij} \quad (13)$$

III. PROPOSED METHOD

This research proposes an optimized method for PayPal REST API functionalities utilizing COPRAS and MOORA, incorporating criteria such as LOCM, SGM, and NOO. The workflow of the proposed method is illustrated in Fig. 1.

A. *Identify Actual Microservices Architecture*

PayPal REST API is taken from the PayPal Developer website, specifically from the Postman collection provided by PayPal.

TABLE I. ACTUAL MICROSERVICE ARCHITECTURE

Microservices	Number of Methods Microservice	Sub Microservices	Number of Methods Sub Microservice
Authorization	4	-	-
Orders	5	-	-
Payments	7	-	-
Invoices	20	Invoices	15
		Templates	5
Subscriptions	20	Catalog products	4
		Plans	7
		Subscriptions	9
Payouts	4	-	-
Webhooks	13	-	-
Shipment Tracking	3	-	-
Transaction Search	2	-	-
Disputes	15	-	-
Manage Accounts	6	-	-
Payment Method Tokens	6	Setup Tokens	2
		Payment token	4

Each microservice definition and usage identified architectures. Its microservice architectures include Authorization, Orders, Payments, Invoices, Subscriptions, Payouts, Webhooks, Shipment Tracking, Transaction Search, Disputes, Onboarding (Limited Release), and Payment Method Tokens. Some microservices have sub-microservices, including Invoices, Subscriptions, and Payment Method Tokens. Each of the microservices has its own methods according to its needs. Table I shows the actual architecture of PayPal REST API.

B. *Propose Optimal Microservices Architecture Alternatives*

This research proposes several optimal alternatives for PayPal REST API microservices architecture. Sub-microservices' potential to become independent microservices leads this research to be the first microservices architecture alternative for PayPal REST API. The first alternative is built by levelling up each sub microservices to independent microservices. The following optimal alternatives are based on the previous alternative with exceptions. The second alternative considers Payment Method Tokens as a non-separatable microservice. The third alternative is based on the second one, which considers the collision potential of extensive microservices usage in refund methods. The Refund method is combined into a separate microservice from Invoices and Payments in the third alternative. The last alternative is based on the third alternative, except the Webhooks microservice, which is divided into two by usage. Webhooks, another large microservice in the PayPal REST API, could be smaller microservices. In the last alternative, Events-related methods in the Webhooks are made into a separate microservice. The proposed optimal microservices architecture is illustrated in Table II.

C. *Evaluate Actual and Proposed Microservice Architectures*

The actual and proposed PayPal REST API microservices architectures were evaluated using MOORA and COPRAS. The architectures were assessed using various weights of optimization criteria through MOORA and COPRAS evaluation. ALCOM, ASGM, and NOO are used as the optimization criteria.

This research focuses on independently reliable microservices architecture. Granularity is the focus of ensuring the scalability of PayPal REST API. Cohesion and complexity are also included in a lower proportion. The weight of ASGM is higher than that of NOO and ALCOM. The weights of NOO and ALCOM are set the same and twice each other in every ASGM weight evaluation.

Table III shows four weight ratio settings for evaluation criteria weights. Set 1 has an equal proportion for all criteria. The other sets have three types of NOO and ALCOM weight ratios: the same portion, twice the ALCOM portion, and twice the NOO portion. Set 2 has a medium weight of granularity with half of all criteria. Meanwhile, set 3 has a high two-thirds weight, and set 4 has a very high three-quarters weight.

D. *Optimize Microservice Architecture*

In optimizing the microservices architecture for the PayPal REST API, the study thoroughly analyses the MOORA and COPRAS rankings to determine the most optimal architecture alternative. Decide the optimal microservice architecture based on each evaluation aim.

TABLE II. ARCHITECTURE ALTERNATIVES COMPARISON

Microservice Architecture	**Actual**	**1st Alternative**	**2nd Alternative**	**3rd Alternative**	**4th Alternative**
Microservices	Authorization	Authorization	Authorization	Authorization	Authorization
	Orders	Orders	Orders	Orders	Orders
	Payments	Payments	Payments	Payments	Payments
				Refund	Refund
	Invoices	Invoices	Invoices		
				Invoices	Invoices
		Templates	Templates	Templates	Templates
	Subscriptions	Catalog products	Catalog products	Catalog products	Catalog products
		Plans	Plans	Plans	Plans
		Subscriptions	Subscriptions	Subscriptions	Subscriptions
	Payouts	Payouts	Payouts	Payouts	Payouts
	Webhooks	Webhooks	Webhooks	Webhooks	Webhooks
					Event
	Shipment Tracking	Shipment Tracking	Shipment Tracking	Shipment Tracking	Shipment Tracking
	Transaction Search	Transaction Search	Transaction Search	Transaction Search	Transaction Search
	Disputes	Disputes	Disputes	Disputes	Disputes
	Manage Accounts	Manage Accounts	Manage Accounts	Manage Accounts	Manage Accounts
	Payment Method Tokens	Vault Payment Source - Setup Tokens	Payment Method Tokens	Payment Method Tokens	Payment Method Tokens
		Vault Payment Source - Payment token			
Total Microservices	12	16	15	16	17

TABLE III. EVALUATION CRITERIA WEIGHTS

Evaluation Set	**Description**	**Weight of Each Criterion**		
		ASGM	**NOO**	**ALCOM**
1	Equals	0.33	0.33	0.33
2	Medium granularity	0.50	0.25	0.25
2a	Medium granularity, better cohesion	0.50	0.17	0.33
2b	Medium granularity, better cohesion	0.50	0.33	0.17
3	High granularity	0.67	0.17	0.17
3a	High granularity, better cohesion	0.67	0.11	0.22
3b	High granularity, better complexity	0.67	0.22	0.11
4	Very high granularity	0.75	0.13	0.13
4a	Very high granularity, better cohesion	0.75	0.08	0.17
4b	Very high granularity, better complexity	0.75	0.17	0.08

IV. Evaluation and Results

In this study, numerical analysis was used to evaluate optimized microservice architectures by calculating key metrics of SGM, NOO, and LOCM. Employing COPRAS and MOORA, the research quantitatively assessed various architectural configurations.

A. *Evaluation*

After ranking the actual and alternative architecture, both MOORA and COPRAS evaluations have the same results in every weight scenario, as illustrated in Tables IV and V. This consistent alignment between MOORA and COPRAS across different scenarios underscores the robustness of the evaluation methods and the reliability of the results.

In evaluation 1, the most optimal architecture is the fourth alternative. The fourth alternative architecture has the highest number of microservices of all alternatives. The architecture has fine grain by dividing actual microservices into smaller ones and considering each usage. The architecture can also maintain balance, complexity, and cohesion.

In evaluation 2, the third alternative architecture is recommended. It is nearly the same as the fourth alternative, except Webhooks remain one whole microservice. By merging Webhooks from the fourth alternative, this architecture becomes more reliable at medium granularity. It suits a variety of complexity and cohesion goals.

In evaluation 3, the second alternative is the most optimal architecture. This architecture promotes actual microservices becoming standalone, except for Payment Method Tokens. It may be the optimal architecture to achieve high granularity.

In evaluation 4, the recommendation varies among different cohesion-complexity scenarios. However, each scenario's ranking is similar. The difference between them is their best and second-best alternative recommendations. The second alternative has the upper hand for a better complexity goal. Meanwhile, the actual architecture best suits the goal of lowering complexity.

B. *Results*

According to the evaluation result in Table VI, almost every alternative has its strength to achieve goals. The fourth alternative has the upper hand in granularity, complexity, and cohesion inequality. The third alternative architecture may suit the medium goal for granularity, complexity, and cohesion. The second alternative can excel in high and very high granularity while considering more complexity. The actual architecture is also suitable for achieving very high granularity but almost ignoring the complexity.

TABLE IV. MOORA Architectures Ranking

PayPal REST API Architecture	**Ranking of MOORA in Each Evaluation**									
	1	***2***	***2a***	***2b***	***3***	***3a***	***3b***	***4***	***4a***	***4b***
Actual	5	5	5	5	3	2	4	**1**	**1**	2
1st Alternative	3	4	4	4	5	5	5	5	5	5
2nd Alternative	4	3	2	3	**1**	**1**	**1**	2	2	**1**
3rd Alternative	2	**1**	**1**	**1**	2	3	2	3	3	3
4th Alternative	**1**	2	3	2	4	4	3	4	4	4

TABLE V. COPRAS Architectures Ranking

PayPal REST API Architecture	**Ranking of COPRAS in Each Evaluation**									
	1	***2***	***2a***	***2b***	***3***	***3a***	***3b***	***4***	***4a***	***4b***
Actual	5	5	5	5	3	2	4	**1**	**1**	2
1st Alternative	3	4	4	4	5	5	5	5	5	5
2nd Alternative	4	3	2	3	**1**	**1**	**1**	2	2	**1**
3rd Alternative	2	**1**	**1**	**1**	2	3	2	3	3	3
4th Alternative	**1**	2	3	2	4	4	3	4	4	4

TABLE VI. Optimum Microservices Architecture Results

Goals in granularity, complexity, cohesion	**Recommended Microservice Architecture**
Equals	4th alternative
Medium granularity	3rd alternative
High granularity	2nd alternative
Very high granularity, better complexity	2nd alternative
Very high granularity, complexity-cohesion equal and better cohesion	actual architecture

V. Conclusion

PayPal REST API microservices architecture can be optimized to other optimal alternatives, considering the API's needs. The microservice architecture optimization should have a designed composition for granularity, complexity, and cohesion.

MCDM can be used to evaluate PayPal REST API alternatives. By proposing some possible optimal microservices architectures, MCDM results from an architecture recommendation for each goal scenario. The architecture can be optimized using the second alternative for high granularity demand. The third alternative may work well for medium granularity. The fourth alternative can also be considered for equal granularity, cohesion, and complexity needs. However, the actual architecture may work better in very high granularity, almost ignoring complexity.

Designing methods for the business goal will be useful in future work. The straightforward goal makes the microservice architecture optimization more reliable in business applications.

Acknowledgment

This research was funded by Institut Teknologi Sepuluh Nopember (ITS) under Penelitian Dana Departemen Program, Penelitian Keilmuan, Penelitian Flagship, Penelitian Kolaborasi Pusat and under the Project Scheme of the Publication Writing and Intellectual Property Rights (IPR) Incentive (Penulisan Publikasi dan Hak Kekayaan Intelektual/PPHKI) Program 2024.

References

[1] S. B. Cleveland, A. Jamthe, S. Padhy, J. Stubbs, M. Packard, J. Looney, S. Terry, R. Cardone, M. Dahan, and G. A. Jacobs, "Tapis API Development with Python: Best Practices In Scientific REST API Implementation: Experience implementing a distributed Stream API," in *Practice and Experience in Advanced Research Computing*, in PEARC '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 181–187. doi: 10.1145/3311790.3396647.

[2] I. Karabey Aksakalli, T. Çelik, A. B. Can, and B. Tekinerdoğan, "Deployment and communication patterns in microservice architectures: A systematic literature review," *Journal of Systems and Software*, vol. 180, p. 111014, 2021, doi: https://doi.org/10.1016/j.jss.2021.111014.

[3] G. Blinowski, A. Ojdowska, and A. Przybyłek, "Monolithic vs. Microservice Architecture: A Performance and Scalability

Evaluation," *IEEE Access*, vol. 10, pp. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.
[4] S. Hassan, R. Bahsoon, and R. Kazman, "Microservice transition and its granularity problem: A systematic mapping study," *Softw Pract Exp*, vol. 50, no. 9, pp. 1651–1681, 2020, doi: https://doi.org/10.1002/spe.2869.
[5] S. Hassan, R. Bahsoon, and R. Buyya, "Systematic scalability analysis for microservices granularity adaptation design decisions," *Softw Pract Exp*, vol. 52, no. 6, pp. 1378–1401, 2022, doi: https://doi.org/10.1002/spe.3069.
[6] E. G. and A. H. and G.-C. C. M. Vera-Rivera Fredy H. and Puerto-Cuadros, "Microservices Backlog - A Model of Granularity Specification and Microservice Identification," in *Services Computing – SCC 2020*, Y. and S. S. and Z. L.-J. Wang Qingyang and Xia, Ed., Cham: Springer International Publishing, 2020, pp. 85–102.
[7] V. Raj and R. Sadam, "Evaluation of SOA-Based Web Services and Microservices Architecture Using Complexity Metrics," *SN Comput Sci*, vol. 2, no. 5, p. 374, 2021, doi: 10.1007/s42979-021-00767-6.
[8] N. Santos and A. Rito Silva, "A Complexity Metric for Microservices Architecture Migration," in *2020 IEEE International Conference on Software Architecture (ICSA)*, 2020, pp. 169–178. doi: 10.1109/ICSA47634.2020.00024.
[9] M.-D. Cojocaru, A. Uta, and A.-M. Oprescu, "Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, 2019, pp. 84–93. doi: 10.1109/ISPDC.2019.00021.
[10] O. Al-Debagy and P. Martinek, "A Metrics Framework for Evaluating Microservices Architecture Designs," *Journal of Web Engineering (JWE)*, vol. 19, pp. 341–370, Apr. 2020, doi: 10.13052/jwe1540-9589.19341.
[11] K. Sellami, A. Ouni, M. A. Saied, S. Bouktif, and M. W. Mkaouer, "Improving microservices extraction using evolutionary search," *Inf Softw Technol*, vol. 151, p. 106996, 2022, doi: https://doi.org/10.1016/j.infsof.2022.106996.
[12] I. Emovon and O. S. Oghenenyerovwho, "Application of MCDM method in material selection for optimal design: A review," *Results in Materials*, vol. 7, p. 100115, 2020, doi: https://doi.org/10.1016/j.rinma.2020.100115.
[13] S. S. Goswami and D. K. Behera, "Solving Material Handling Equipment Selection Problems in an Industry with the Help of Entropy Integrated COPRAS and ARAS MCDM techniques," *Process Integration and Optimization for Sustainability*, vol. 5, no. 4, pp. 947–973, 2021, doi: 10.1007/s41660-021-00192-5.
[14] R. G. Willem Karel M. Brauers and V. Podvezko, "Regional development in Lithuania considering multiple objectives by the MOORA method," *Ukio Technologinis ir Ekonominis Vystymas*, vol. 16, no. 4, pp. 613–640, 2010, doi: 10.3846/tede.2010.38.
[15] B. Goossens, "Decision-Making in a Microservice Architecture." Nov. 2019. [Online]. Available: http://essay.utwente.nl/80113/
[16] H. Taherdoost and M. Madanchian, "Multi-Criteria Decision Making (MCDM) Methods and Concepts," *Encyclopedia*, vol. 3, no. 1, pp. 77–87, 2023, doi: 10.3390/encyclopedia3010006.
[17] C. Y. Sakti, K. R. Sungkono, and R. Sarno, "Determination of Hospital Rank by Using Analytic Hierarchy Process (AHP) and Multi Objective Optimization on the Basis of Ratio Analysis (MOORA)," in *2019 International Seminar on Application for Technology of Information and Communication (iSemantic)*, IEEE, 2019, pp. 178–183.
[18] A. S. Ajrina, R. Sarno, and R. V Hari Ginardi, "Comparison Of MOORA and COPRAS Methods Based on Geographic Information System For Determining Potential Zone of Pasir Batu Mining," in *2019 International Conference on Information and Communications Technology (ICOIACT)*, 2019, pp. 360–365. doi: 10.1109/ICOIACT46704.2019.8938465.
[19] R. P. Setyono and R. Sarno, "Comparative Method of Moora and Copras Based on Weighting of the Best Worst Method in Supplier Selection at ABC Mining Companies in Indonesia."
[20] A. Wilta and A. I. Kistijantoro, "Automatic Measurement of Microservice Architecture Quality with Cohesion, Coupling, and Complexity Metrics," in *2023 10th International Conference on Advanced Informatics: Concept, Theory and Application (ICAICTA)*, 2023, pp. 1–6. doi: 10.1109/ICAICTA59291.2023.10390525.
[21] E. N. H. Kirğil and T. E. Ayyildiz, "Analysis of Lack of Cohesion in Methods (LCOM): A Case Study," in *2021 2nd International Informatics and Software Engineering Conference (IISEC)*, 2021, pp. 1–4. doi: 10.1109/IISEC54230.2021.9672419.
[22] S. Alahmari, E. Zaluska, and D. C. De Roure, "A Metrics Framework for Evaluating SOA Service Granularity," in *2011 IEEE International Conference on Services Computing*, 2011, pp. 512–519. doi: 10.1109/SCC.2011.98.
[23] O. Al-Debagy and P. Martinek, "Extracting Microservices' Candidates from Monolithic Applications: Interface Analysis and Evaluation Metrics Approach," in *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, 2020, pp. 289–294. doi: 10.1109/SoSE50414.2020.9130466.
[24] N. Karakaşoğlu Kundakcı, "Combined Multi-Criteria Decision Making Approach Based On Macbeth And Multi-MOORA Methods," *Alphanumeric Journal*, vol. 4, Apr. 2016, doi: 10.17093/aj.2016.4.1.5000178402.
[25] S. Sutarno, M. Mesran, S. Supriyanto, Y. Yuliana, and A. Dewi, "Implementation of Multi-Objective Optimazation on the Base of Ratio Analysis (MOORA) in Improving Support for Decision on Sales Location Determination," *J Phys Conf Ser*, vol. 1424, no. 1, p. 12019, Dec. 2019, doi: 10.1088/1742-6596/1424/1/012019.
[26] K. R. Sungkono, T. A. Dewa Prakasa, A. R. Hermawan, G. A. Bhamakerti, and R. Sarno, "Optimizing Time and Cost Activity based on Discrete Event Simulation with Multi-Objective Optimization Method by Ratio Analysis (MOORA)," in *2022 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2022, pp. 53–58. doi: 10.1109/ICACSIS56558.2022.9923512.
[27] A. S. Ajrina, R. Sarno, H. Ginardi, and A. Fajar, "Mining zone determination of natural sandy gravel using fuzzy AHP and SAW, MOORA and COPRAS methods," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 5, pp. 560–571, Oct. 2020, doi: 10.22266/ijies2020.1031.49.
[28] Y. Kustiyahningsih, Husni, and I. Q. Aini, "Integration of FAHP and COPRAS Method for New Student Admission Decision Making," in *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, 2020, pp. 1–6. doi: 10.1109/ICVEE50212.2020.9243260.