Microservice-Driven Modular Low-Code Platform for Accelerating SME Digital Transformation

Zhiwen Fang *

Posted Date: 4 June 2025

doi: 10.20944/preprints202506.0279.v1

Keywords: Microservices; Low-Code Platform; Digital Transformation; Multi-Tenant Security

Article

Microservice-Driven Modular Low-Code Platform for Accelerating SME Digital Transformation

Zhiwen Fang

Department of Information Technology and Management, Illinois Institute of Technology, Chicago, USA; zfang12@hawk.iit.edu

Abstract: To overcome the difficulties encountered by the SMEs during their digital transformation such as insufficient development resources and complicated system integration, this article introduces a microservice-oriented modular low code platform (MDMLCP) to facilitate the rapid devolution and smart evolution of SMEs' digital infrastructure. Specifically, the proposed model embeds new the multi-tenant secure software development lifecycle to guarantee security and privacy between different organizations via data isolation and access control. Meanwhile, the platform adds AI-supported module development functions and integrates the AutoML tool chain, enabling users from small and medium-sized enterprise to develop intelligent applications, such as sales forecasting, inventory optimization and customer portraits, with no IT experience required. Moreover, MDMLCP offers handy digital service templates CRM, ERP and supply chain management systems, enabling the templatized customization of a tailored-for-user business system fast through visual drag and drop and smart connectors. Compared with prior low-code platforms, the MDMLCP creatively presents a meta-component orchestration engine that combines static template reuse and dynamically synthesized AI components to realize intelligent-inspired combination on the component level and the secure collaboration at service level, and is responsible for unified schedule as well as controls over a cloud microservice cluster. The experimental results indicate that MDMLCP can not only raise the system deployment efficiency by 40% and promote the service stability by 35% on average under realistic DSME.

Keywords: microservices; low-code platform; digital transformation; multi-tenant security

1. Introduction

In recent years, the upsurge of worldwide digital transformation has affected all sectors. This trend has gradually transformed into a strategic option for companies to enhance their core competitiveness and business resilience. Business models, customer relationships and organizational processes are being fundamentally transformed by digital technologies. Large companies have long been advanced in automating operations, making intelligent decisions, and data-driven management, rich in capital investment, advanced IT team, and mature digital strategy, but SMES still face many obstructions to the digital transformation due to the structure restrictions [1].

The emergence of low-code development platforms offers an effective way for SMEs to overcome the digital divide. The low-code platform is significantly reducing the threshold for non-technical personnel to participate in the work of system building, thanks to such a number of visual interfaces, modular components and drag-and-drop process configuration, thus shortening the cycle of development and saving the labor costs [2]. But most mainstream low code platforms on the market at present are heavy monoliths, without micro-service support, lack of foundation/infrastructure to support the development, and it's hard to adapt to the real business frequent changes, dynamic system expansion, services fine-grained combination, especially in the complex process of automation, subtle combined application scenario, and multi-role collaboration development [3].

 doi:10.20944/preprints202506.0279.v1

In this regard, an agile, cloud-native system architecture is extremely important, especially for small and mid-sized businesses. Under the on-premise model, enterprises need to invest a lot in hardware and software and later for their day-to-day system maintenance and support; for minor players with poor cash flows and limited technical resources it is really a "pain" [4]. On the other hand, cloud native microservice architecture is naturally capable of elastic scaling, high availability and decoupled deployment, it can be allocated dynamically based on the actual business pressure.

When the system is constructed, SMEs venture need an urgent security and stable SDLC management mechanism. In the multi-tenant scenario, the resources of a unified platform are shared by multiple enterprises, and once the access control, data encryption and tenant isolation mechanism are not perfect, it is easy to lead to the problem of information leakage and compliance risk [5]. Consequently, whole-process assurance that every tenant's data space is isolated from each other, and user permissions are strictly controlled, development complies with security specifications (such as, SSDLC) has become a prerequisite for low-code platforms to transform towards industrial applications. But up to now, low code products are generally lack of the designs of security mechanism, and they are unable to meet the applications in the industry demands the multi-tenant and compliance requirements for operations [6].

2. Related Work

Low-code/no-code (LNC) platforms have in recent years become very significant to SME digital transformation. Domański et al. (2023) [7] through a grey correlation analysis, reviewed the LNC platforms offered on the Polish market; the study compared those platforms along the lines of their ease of use, flexibility, potential, service support, cost-effectiveness taking into account the aforementioned eight platforms, the authors affirmed that LNC platforms can indeed help in enhancing the management efficiency of SMEs, especially in the face of limited resources or poor IT capacity. Mihu (2022) [8] studies low-code platforms' contribution to the digital transformation of enterprises, noting that both the development cycle of application and the enterprise agility have been increased' reduced relying on professionalized developers for low-code development, as well as the reminder to enterprises to pay attention to the platform scalability, platform integration with the existing system when introducing the low-code platform, in order to avoid future technical debt.

Bies et al. (2022) [9] adopt a mixed method to explore the driving effects of low-code development platforms on the digital innovation of SMEs and discovered that low-code platforms can enlarge the digital innovation capabilities of SMEs, especially by making applications to develop and deploy faster, business operations to be more flexible, and responses to customer to be more real-time. Cai et al. (2022) [10] proposed a six-step low-code implementation process, from process modeling, over platform selection up to deployment evaluation. From practical application cases, the method has great advantages in cost reduction, flexibility enhancement and implementation cycle shortening for SMEs with few resources and urgent process optimization requirement.

Liu and Cui (2023) [11] discuss the application of low-code technology in intelligent transportation systems. This study proposes a solar-powered on-board hopping card control system developed using a low-code platform, aiming to improve the hopping control efficiency of public transport vehicles at specific stations. Through the visual programming and modular design of the low-code platform, the system realizes rapid deployment and flexible configuration.

3. Methodologies

3.1. Cloud-Based Multi-Tier Microservice Architecture

In the basic design of this platform, we first built a multi-layer microservice architecture based on cloud computing to achieve elastic deployment and unified scheduling of various low-code components in the cloud. The core goal of this architecture is to ensure that SMEs can not only efficiently reuse basic capabilities. Suppose platform manages n microservice units, each microservice is represented as Equation 1:

 doi:10.20944/preprints202506.0279.v1

$$S_i =< \phi_i, \lambda_i, \delta_i >, \tag{1}$$

Specifically, ϕ_i represents the functional semantic mapping of the service, such as "sales forecasting" or "customer management", which is used to connect business needs with technical implementation. λ_i represents the resource consumption function of the microservice, which involves the quantitative calculation of physical resources such as CPU, memory, and bandwidth, and δ_i represents the deployment strategy function of the service, which is used to guide its scheduling and auto scaling logic in the microservice cluster.

In order to achieve global optimal scheduling of resources at the platform layer, we introduce the following objective functions to minimize the overall resource occupation while maintaining high service availability, as shown in Equation 2:

$$\min_x \sum_{i=1}^{n} \lambda_i(x_i) - \alpha \cdot \sum_{i=1}^{n} A_i(x_i), \tag{2}$$

where $x_i \in \{0,1\}$ indicates whether first i service is deployed (1 is deployed, 0 is not deployed). $A_i(x_i)$ score for the availability of the service under the current deployment configuration. α is a positive moderator that weighs the relationship between resource conservation and availability guarantees. To ensure that the platform does not exceed the resource quota in the current cloud environment, the following constraints are further introduced, as Equation 3:

$$\sum_{i=1}^{n} \lambda_i(x_i) \leq R_{max}, \qquad \forall x_i \in \{0,1\}, \tag{3}$$

where R_{max} indicates the total resource limit, such as the total amount of memory currently available on the platform and the total number of CPU cores. In addition to resource scheduling, the platform also focuses on introducing the multi-tenant secure software development life cycle mechanism to ensure that the tenant set of the current service of the isolation and installation system between enterprises is $T = \{T_1, T_2, \ldots, T_m\}$, and each tenant runs in logical isolation container, as Equation 4:

$$C_j =< D_j, P_j, A_j >, \tag{4}$$

where D_j is the private dataset of tenant T_j, which must be ensured not to be accessed by other tenants. P_j is the access policy matrix, which is used to define the authorization logic between users, objects, and operations. A_j is the collection of all business applications deployed by the tenant. We use the attribute-based access control (ABAC) mechanism to model inter-tenant access to resources. The authorization decision function is as follows in Equation 5:

$$Permit(u, o, a) \Leftrightarrow \exists \theta \in \Theta: f_u(u) \wedge f_o(o) \wedge f_a(a) \Longrightarrow True, \tag{5}$$

where u is the user entity, o is the object to be accessed, a is the operation behavior, and f_* is the attribute function extracted from the user, object, and operation respectively. Θ is a predefined policy rule set. To ensure tenant data isolation, the data overlap rate between tenants is introduced, as Equation 6:

$$\eta_{ij} = \frac{|D_i \cap D_j|}{\min(|D_i|, |D_j|)} \; when \; \eta_{ij} \approx 0, \forall i \neq j, \tag{6}$$

where D_i and D_j are datasets for two tenants, respectively. This metric is used to measure the degree of intersection between two tenants, and the platform uses container isolation, sandbox protection, and cryptographic hashing mechanisms to ensure that the value is always close to 0 and that the boundaries between enterprises are strictly demarcated.

In terms of access control, the platform carries out permission authorization and policy enforcement based on the attribute-based access control model. The model constructs an access request triplet based on user attributes, object attributes and behavior attributes, and matches them

 doi:10.20944/preprints202506.0279.v1

with the preset policy rules of the system, and only the combination of policy expressions is allowed to access resources.

3.2. Low-Code Construction and Intelligent Orchestration

The platform innovatively introduces the AutoML module to assist SMB users in quickly generating pluggable intelligent service components, such as sales forecasting, inventory optimization, etc. Users only need to configure input and output variables in the graphical interface, and the platform can generate optimized machine learning pipelines through automatic modeling and parameter tuning. Let the original sample dataset be Equation 7:

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}, \tag{7}$$

where $x_i \in \mathbb{R}^d$ is the input eigenvector and $y_i \in \mathbb{R}$ is the target value. The goal of AutoML is to automatically select the optimal model from the candidate model set $\mathcal{M}$ and optimize its parameter θ so that the model has the smallest loss function on a given data, as Equation 8:

$$\arg \min_{M \in \mathcal{M}, \theta} \mathcal{L}(M(x; \theta), y) + \lambda \cdot \Omega(M), \tag{8}$$

where $\mathcal{L}$ is the loss function, such as MSE or cross-entropy, $\Omega(M)$ is the model complexity regularization term, and λ is the regularization weight. AutoML searches for structures and hyperparameters through structure search algorithms (such as Bayesian optimization, reinforcement learning, etc.) to make the final model optimal. Each meta-component is modeled as a five-tuple as Equation 9:

$$\mathcal{M}_c = < F_c, R_c, I_c, O_c, \Psi_c >, \tag{9}$$

where F_c stands for a feature description. R_c denotes resource requirements; I_c and O_c are the types of input and output interfaces (e.g., input sales data, output forecast results). Ψ_c is the orchestration condition and dependencies of the current component. The goal of the system is to construct an optimal path that satisfies the input-output constraints, as Equation 10:

$$\min_{P \sqsubseteq G} \sum_{e \in P} w(e), s.t. IO - compatibility(P) = True, \tag{10}$$

where P is the combination of candidate service paths, and the system uses dynamic programming and topology sorting to ensure data interface matching. To support model versioning and reuse, MDMLCP supports model caching and automatic retraining mechanisms of AutoML components. After the data is updated, the platform can automatically retrain the corresponding model according to the threshold to ensure that the model always adapts to business changes and supports.

4. Experiments

4.1. Experimental Setup

The online retail dataset that was recorded on the UCI Machine Learning Repository, the real-world experimental dataset is employed, which contains the records of an online retailer in the UK from 2010 to 2011, accounting for about 500, 000 transaction records, which involves multi-perspective information such as customer behavior, products sales and time and so on, and it is typical and has the practical applicability. Through the importation of data to the MDMLCP platform, a complete AutoML based sales forecast application is constructed, and it covers data cleaning, model selection, predicting result visualization, which fully verify the intelligent construction ability and efficient delivery advantage to the small and medium-sized enterprise scenario of the platform.

Deployment efficiency improvement refers to the overall time and number of manual interactions from data import to system launch. Service stability is evaluated by the frequency of service interruptions and the success rate of requests recorded during 7 days of continuous operation. In order to further verify the versatility and practicability of MDMLCP, this study expands the scope of experimental evaluation and adds two typical application scenarios: rapid development and

 doi:10.20944/preprints202506.0279.v1

deployment of CRM (customer relationship management) and ERP (enterprise resource planning) modules. Based on the platform's built-in digital service templates, users can build applications that include customer data processing, inventory management and other functions by simply using graphical drag-and-drop methods. To fully evaluate the holistic performance of the proposed MDMLCP framework in DT of SMEs, four representative comparative approaches as baselines:

- OutSystems (OS) is a main-stream commercial low-code platform with strong front-end and back-end integration, but no built-in AI support;
- Mendix (MDX) takes on the mdd as the core for suitable for rule logic construction, but need to rely on the external tools with intelligent modeling;
- Google AutoML-GCP(AutoML) with the built-in AutoML module are adopted for comparing the modeling accuracy and integration efficiency with high-accuracy automatic modeling capabilities from AutoML.
- AppGyver combined with Firebase (AGF) is more like a light weight no-code solution appropriate for fast prototyping, we compared the differences in component reusability and system scalability.

4.2. Experimental Analysis

The prediction accuracy measures the ability of a model to make predictions on a test set. As shown in Figure 1, there has a trend that the prediction accuracy of each method is able to converge into a saturation growth for faster and plateau growth at point prediction quality along an increasing sample size and is conformable to the regulation where the machine learning model performance degrades with a rising of the sample number.

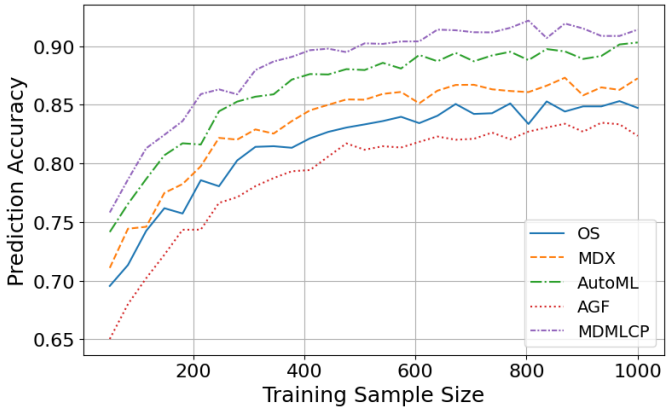

Figure 1. Prediction Accuracy Comparison.

More precisely, the curve of MDMLCP always stands in the first place, reaching 0.80 of precision when the sample size is small and stabilizing at 0.92 in the large sample stage, which suggests its automatic ML and microservice collaborative mechanism can offer great prediction performance under both shortage and plenty of data circumstances. Service response latency is a measure of the time it takes for a microservice architecture to respond to a front-end request in a cloud deployment environment.

As the number of concurrent requests increases from 1 to 200 of Figure 2, the response latency of each platform shows a significant upward trend, but there are significant differences in the increase amplitude. MDMLCP consistently maintains the lowest latency across all concurrency levels approximately 350ms even under 200 concurrent requests, while other methods have exceeded 400 ms, indicating that MDMLCP's microservice splitting and metacomponent orchestration engine can still efficiently schedule resources and reduce request waiting under high load. In contrast, the response performance of traditional low-code platforms and external AutoML and no-code solutions is more significant when the concurrency is under pressure.

 doi:10.20944/preprints202506.0279.v1

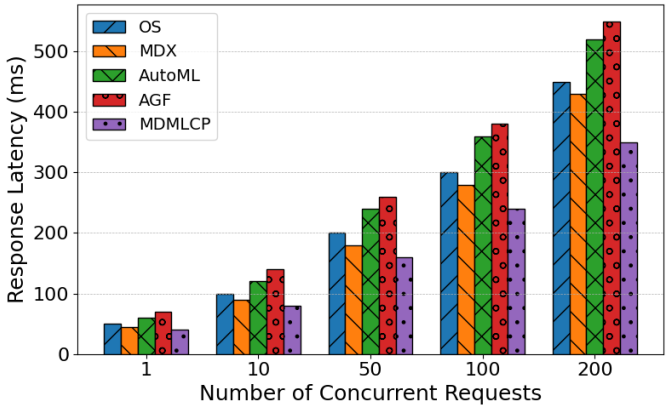

Figure 2. Service Response Latency Comparison with Hatch Patterns.

The platform encapsulates AutoML modeling results into visual "smart components" and automatically registers them with the Service Bus for drag-and-drop invocation in business processes. As shown in Figure 3, all three core components of MDMLCP significantly impact overall performance. Removing AutoML reduces prediction accuracy to 87%, while disabling the orchestration engine and meta-component mechanism lowers deployment efficiency to 78% and 83%, respectively, underscoring their roles in intelligent modeling.

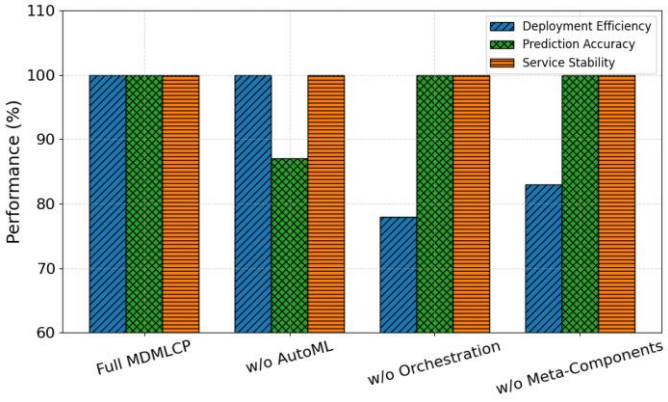

Figure 3. Ablation Results on MDMLCP Core Modules.

5. Conclusion

In conclusion, the microservice-driven modular low-code platform proposed in this paper realizes the rapid delivery and intelligent upgrade of SMEs in the construction of key business systems such as CRM, ERP, and supply chain through the deep integration of cloud-native microservice architecture, multi-tenant security development lifecycle, intelligent component generation, and orchestration engine. Experimental results show that MDMLCP has significant advantages over mainstream low-code/no-code solutions in terms of deployment efficiency, prediction accuracy, and service response delay. As for future, expand the AI module library to explore automated anomaly detection and decision optimization capabilities. It also strengthens permission and compliance management based on the zero-trust security model to help MDMLCP achieve resilient and intelligent digital transformation in industry scenarios. Specifically, it includes dynamic identity verification, behavioral mode access control, and real-time risk assessment mechanisms to achieve a higher level of data security and compliance assurance, and help the platform achieve digital transformation in industrial scenarios.

 doi:10.20944/preprints202506.0279.v1

References

1. i Palomés, X. P., Tuset-Peiró, P., & i Casas, P. F. (2021, October). Combining low-code programming and sdl-based modeling with snap! in the industry 4.0 context. In 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) (pp. 741-750). IEEE.
2. Xiaosheng, J., Hua, Z., Jin, Z., AL-Bukhaiti, K., & Wan, A. (2025). Overcoming digital transformation barriers: Chinese SMMEs in the sharing economy. Frontiers in Manufacturing Technology, 5, 1495840.
3. Marek, K., Śmiałek, M., Rybiński, K., Roszczyk, R., & Wdowiak, M. (2021). BalticLSC: Low-code software development platform for large scale computations. Computing and Informatics, 40(4), 734-753.
4. Kirchhoff, J., Weidmann, N., Sauer, S., & Engels, G. (2022, October). Situational development of low-code applications in manufacturing companies. In Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (pp. 816-825).
5. Overeem, M., Jansen, S., & Mathijssen, M. (2021, June). API management maturity of low-code development platforms. In International Conference on Business Process Modeling, Development and Support (pp. 380-394). Cham: Springer International Publishing.
6. Wang, P., Wang, Z., Duan, J., Miao, Y., & Tian, W. (2025). Review of the development status of low-code programming software for robots. Digital Engineering, 100039.
7. Domański, R., Wojciechowski, H., Lewandowicz, J., & Hadaś, Ł. (2023). Digitalization of Management Processes in Small and Medium-Sized Enterprises—An Overview of Low-Code and No-Code Platforms. Applied Sciences, 13(24), 13078.
8. Mihu, C. (2022). The Low-Code Movement: Accelerating Digital Transformation with Low-Code. In Handbook of Research on Digital Transformation Management and Tools (pp. 556-571). IGI Global.
9. Bies, L., Weber, M., Greff, T., & Werth, D. (2022, November). A mixed-methods study of low-code development platforms: drivers of digital innovation in SMEs. In 2022 international conference on electrical, computer, communications and mechatronics engineering (ICECCME) (pp. 1-6). IEEE.
10. Cai, F. Z., Huang, S. Y., Kessler, T. S., & Fottner, F. J. (2022). A case study: Digitalization of business processes of SMEs with low-code method. IFAC-PapersOnLine, 55(10), 1840-1845.
11. Liu, G., & Cui, Z. (2023, November). Solar Carrier Skipping Station Card Control Based on Low-Code Platforms. In Proceedings of the 2023 5th International Conference on Internet of Things, Automation and Artificial Intelligence (pp. 318-323).