International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 5, Issue 3, 142-147, 2024
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V5I3P115

Original Article

Low-Code/No-Code DevOps Automation for Accelerated Digital Transformation in Global Enterprises

Pankaj Gupta
Independent Researcher, USA.

Abstract *- With an ever-increasing number of global corporations utilizing an exponentially expanding digital footprint, it is becoming increasingly important to evolve how software is deployed. Organizations with large application portfolios, geographically distributed teams, and hybrid or multi-cloud environments are continuously challenged to improve delivery speed while maintaining high standards of security, governance, and regulatory compliance. Although traditional DevOps automation approaches based on scripting, manual configuration, and specialized technical expertise have improved collaboration and deployment speed, they often limit scalability and restrict broader organizational participation due to their reliance on advanced technical skills. This paper presents an analysis and conceptual design of an enterprise-oriented DevOps automation approach based on the low-code and no-code (LC/NC) paradigm. The research examines whether LC/NC DevOps automation can enable more scalable and standardized software delivery workflows across large organizations. The proposed approach introduces reusable, visual, and model-driven components to abstract pipeline orchestration, infrastructure provisioning, and operational processes within DevOps automation. These abstractions allow teams and business units to execute DevOps activities asynchronously through self-service mechanisms, independent of centralized engineering teams. The conceptual framework incorporates several key architectural features, including standardized pipeline templates to support scalability, policy-driven governance to ensure security and compliance, role-based access control to manage participation, and integrated observability to monitor pipeline execution and system behavior. Analysis of existing literature and enterprise DevOps practices suggests that LC/NC DevOps automation can improve delivery speed, reduce operational complexity, and enhance scalability when compared with traditional DevOps implementations. Overall, the findings indicate that enterprise-grade LC/NC DevOps automation provides a practical and sustainable foundation for long-term digital transformation, operational resilience, and continuous innovation.*

Keywords *- Low-Code Platforms, No-Code Automation, DevOps Automation, Digital Transformation, Enterprise IT, Enterprise DevOps Automation, Pipeline-As-Code Abstraction, Policy-Driven DevOps Governance.*

1. Introduction

Companies today are involved in developing large, distributed software systems that are continuously changing to meet emerging needs. Most companies will utilize numerous applications for their business operations, customer relationships, and internal operations. The development teams are located globally and now utilize various cloud services to support their technology environment. Therefore, the challenge of deploying reliable software quickly has increased; furthermore, companies are subject to strict regulations, security requirements, and governance requirements.

To address these challenges, many organizations have begun to adopt DevOps methodologies. The goal of DevOps is to improve communication and collaboration between development and operations teams, and to increase the efficiency of automated software delivery through methods such as continuous integration and continuous delivery (CI/CD). By utilizing DevOps, many organizations have experienced an increase in the frequency of releasing software, as well as improved system stability. However, for larger organizations, the automation of DevOps is commonly performed through the utilization of complex scripts and manually defined pipelines. The automation of DevOps requires specialized technical knowledge and is generally managed by a few experts.

As organizations continue to grow, the management of automation pipelines becomes increasingly difficult, adding additional time for onboarding new teams, and changes to the delivery process typically require the involvement of a centralized engineering team. Additionally, the compliance and security requirements of an organization may cause the deployment pipeline to be tightly controlled. While this provides a level of oversight, it may slow down the delivery process and limit the ability of teams to make decisions independently. Therefore, many organizations struggle to find a balance between the speed of delivery and the ability to provide consistent control.

A significant area of growth in reducing the complexity associated with software development is the emergence of low-code and no-code (LC/NC) platforms. LC/NC platforms utilize model-based and visual approaches to represent application logic and workflow, requiring minimal amounts of hand-written code. LC/NC platforms have accelerated the delivery of

applications and expanded the number of participants in software creation. It is therefore reasonable to assume that applying these concepts to the automation of DevOps is a natural extension.

In the context of DevOps, low-code and no-code automation enables the visualization of pipeline setup, infrastructure provisioning, and task execution as reusable visual elements. Rather than writing scripts from scratch, teams assemble DevOps workflows using pre-defined building blocks. This methodology makes automation more accessible, adaptable, and reusable for those who do not possess a high level of expertise in DevOps. For global companies, low-code and no-code DevOps automation allows companies to utilize consistent delivery methodologies across teams and geographies while ensuring compliance, security, and control.

Although there is increasing interest in LC/NC platforms, much of the research and guidance available currently focuses on individual tools or limited use cases. There is significantly less guidance provided for implementing and managing low-code and no-code DevOps automation at scale within large enterprise cloud environments. The purpose of this paper is to consolidate current knowledge about LC/NC platforms and DevOps automation, propose a framework for the implementation of low-code and no-code DevOps automation for large enterprises, and outline potential benefits, limitations, and areas of study for future research.

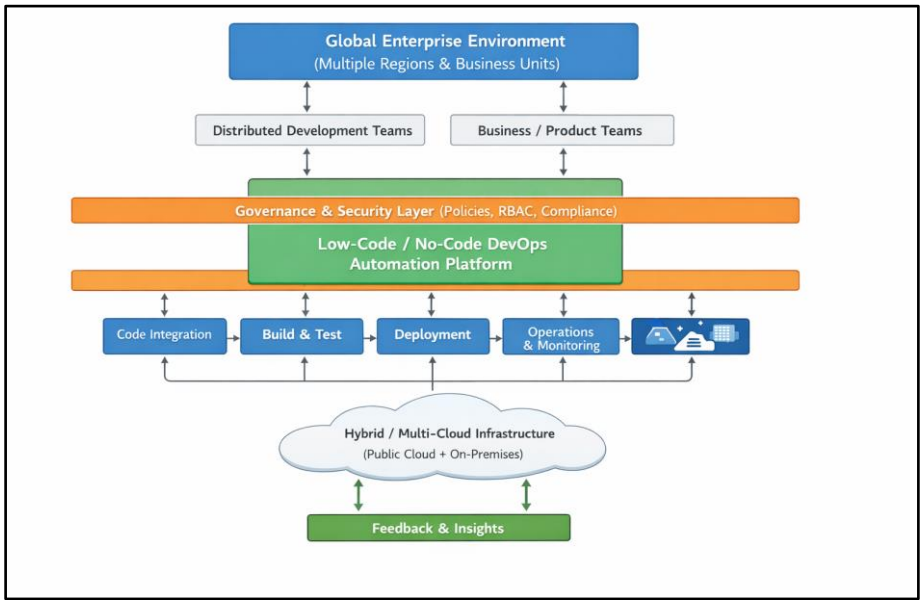

Fig 1: High-Level Overview of Low-Code/No-Code DevOps Automation in Global Enterprises

2. Related Work and Literature Review

The vast majority of prior research related to Low-Code/No-Code (LC/NC) platforms has focused on Rapid Application Development (RAD), Business Process Automation (BPA), and Citizen Development. The literature indicates that these platforms can significantly decrease development time and enable better communication between business and IT groups through the abstraction of technical complexities using visual models or other forms of modeling. These platforms have been shown to increase an organization's agility, particularly in those environments where there is a need for digital solutions that exceeds the availability of developer resources to build them.

Automation of the DevOps life cycle is heavily dependent upon CI/CD pipelines, IaC (infrastructure-as-code), and CM (configuration management) to improve the frequency of deployments and reliability of systems. Research has indicated, however, that the implementation of traditional DevOps processes require significant engineering efforts to design, implement, and scale automation pipelines, potentially hindering their use in large enterprises.

Enterprise governance and security issues remain long-standing barriers to the adoption of LC/NC platforms. While researchers emphasize the importance of developing and implementing policies, access controls based on roles, and auditability to ensure that democratization of automation is compliant with regulatory requirements, they also acknowledge that the integration of LC/NC DevOps platforms with cloud-native services is a key factor to achieve scalable and resilient automation in large-scale enterprise environments.

Recent systematic reviews of LC/NC platforms have highlighted many of the same advantages associated with their use, including lower development costs, improved collaboration between business and IT groups, as well as disadvantages, including platform fragmentation, vendor lock-in, and higher governance costs. As LC/NC platforms continue to be adopted,

organizations are experiencing difficulty in establishing consistent security practices, data management practices, and lifecycle governance across a variety of LC/NC platforms.

In addition to the empirical evidence from case study research and conceptual analysis, the literature also identifies scaling and performance limitations when complex integrations are made and/or high-volume workloads are processed using LC/NC applications. These include limitations related to abstraction overhead, the degree to which customizations can be performed, and the lack of transparency into the generation of artifacts; all of which raise concerns about the suitability of LC/NC platforms for mission-critical, enterprise-wide automation. The findings presented above illustrate the need for enterprise-grade LC/NC DevOps frameworks that explicitly provide for governance, security, and scalability.

Table 1: Comparison of Existing low-code/no-Code Platforms and The Proposed Enterprise LC/NC DevOps Framework

Aspect	Existing LC/NC Platforms	Proposed LC/NC DevOps Framework
Primary Scope	Application and workflow development	End-to-end enterprise DevOps automation
DevOps Coverage	Partial or tool-specific automation	Complete CI/CD and operations lifecycle
Scalability	Limited for large, complex environments	Designed for global, multi-team enterprises
Governance	External or reactive controls	Built-in policy-as-code and RBAC
Security Integration	Post-development or optional	Embedded shift-left security
Cloud Support	Vendor-dependent	Hybrid and multi-cloud abstraction
Operational Visibility	Limited monitoring	Integrated observability and feedback
Enterprise Adoption	Fragmented and inconsistent	Standardized, enterprise-ready platform

3. Methodology and Proposed System

A Design Science Research (DSR) methodology combining conceptual and analytical methods will be employed to develop and assess an enterprise-level Low-Code/No-Code (LC/NC) DevOps automation framework. In contrast to other methodologies, which emphasize specific implementations of commercial tools, DSR will focus on reusable architectural designs, governance frameworks, and workflow abstraction that can be applied across various industries and organizational settings. The methodology will draw upon previous literature on LC/NC platforms, Enterprise DevOps practices, and scalability and governance issues associated with large organizations.

3.1. Conceptual Model

The proposed framework is composed of four interconnected layers - Interface, Orchestration, Governance, and Execution. A layered architecture is used to create modularity; thus, if a change is made to one layer, it should have little impact on the other layers. Layered architectures enable scaling, maintainability, and long-term evolutionary growth within the enterprise environment.

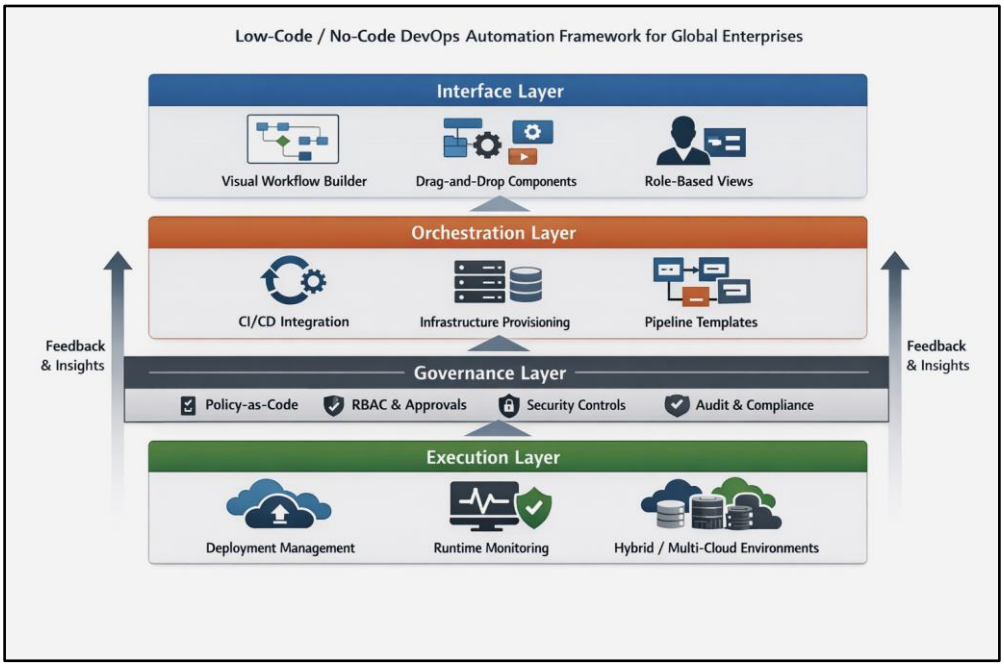

Fig 2: Conceptual low-code/no-code DevOps automation framework for global enterprises

The interface layer provides visual modeling tools that abstract complex DevOps activities into drag-and-drop components and form-based configurations. Users may choose from a catalog of pre-existing elements that represent tasks such as code integration, automated testing, deployment strategies (i.e., Blue-Green and Canary Deployments), and monitoring dashboards. For purposes of addressing the needs of users with differing roles, the interface layer has provided role-specific views. Advanced customization options will be available to professional developers; however, business analysts and product owners will be able to interact with the framework using guided configuration workflows. Visual representations of the workflow will decrease cognitive load, increase the speed of onboarding, and increase consistency across dispersed teams.

The orchestration layer acts as a translation engine, translating the visual representation of the pipeline into an executable workflow. The orchestration layer interfaces with existing Continuous Integration/Continuous Deployment (CI/CD) platforms, Infrastructure-as-Code (IaC) tools, and Cloud Service Application Programming Interfaces (APIs); including Jenkins, GitLab CI, Terraform, and Kubernetes. Standardized pipeline templates act as reusable blueprints for common application types and include best practices such as automated rollback mechanisms, environment promotion gates, and parallel execution. The use of parameterization allows for controlled customization of standardized pipeline templates for the purpose of adapting them to application-specific or regional requirements (e.g., data residency constraints) and adhering to governance standards.

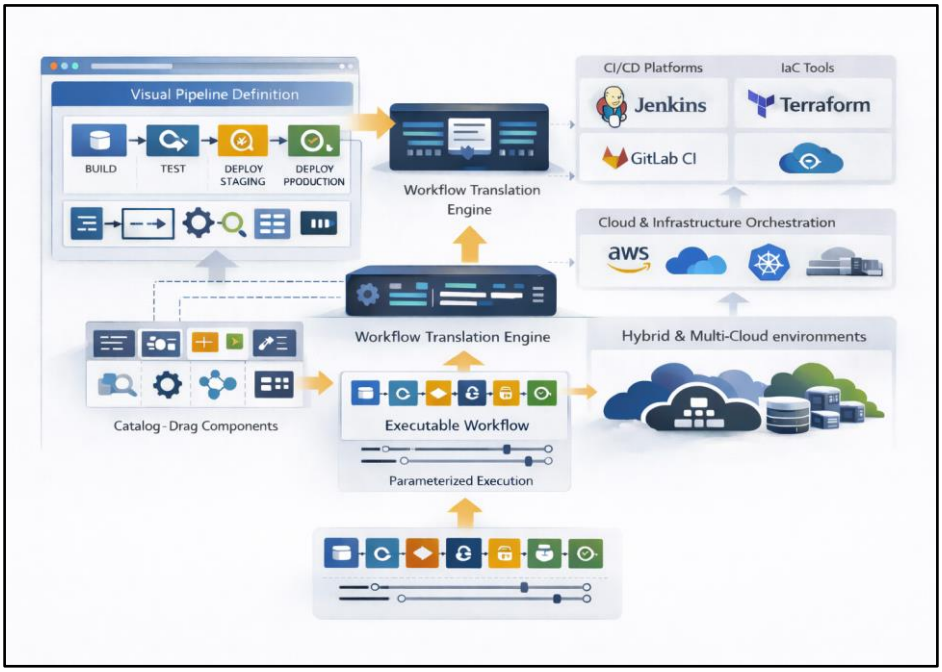

Fig 3: Orchestration Layer Translating Visual Pipelines into Executable Workflows across CI/CD and Cloud Platforms

3.2. Governance and Security Mechanisms

In order for enterprises to successfully adopt LC/NC DevOps automation, they require governance mechanisms that provide a means of balancing the need for self-service agility with centralized oversight. The governance layer enforces policy-as-code mechanisms to govern lifecycle events pertaining to pipeline creation, modification, execution, and decommissioning. Policies are declared and enforced constraints pertaining to access to environments, regulatory compliance, and minimum security standards.

Role-based access control is used to differentiate permission levels for different user categories. Citizen developers typically have limited access to sandbox environments; platform engineers oversee shared templates; and security or compliance teams approve high-risk changes. Approval workflows embedded in the DevOps lifecycle and automated audit logging provide traceability and accountability throughout the DevOps lifecycle. Security practices are integrated early into the DevOps workflow by embedding vulnerability scanning, static analysis, and configuration validation within pipeline templates, thereby preventing the execution of non-compliant pipelines. In order to prevent Shadow IT, the Governance Layer maintains a central inventory of registered pipelines and automates the discovery of unmanaged automations.

3.3. Execution and Integration

The execution layer operationalizes the DevOps workflows by interacting with the runtime environments in development, staging, and production. The execution layer supports hybrid and multi-cloud strategies through abstraction mechanisms that standardize provider-specific APIs, thereby enabling the deployment of applications across public cloud platforms and on-premises Kubernetes environments while minimizing vendor lock-in.

Resilience patterns (e.g., blue-green deployments, traffic shifting canaries, and circuit breakers) are incorporated into execution workflows to mitigate failure risks. Compute-intensive tasks are delegated to optimized microservices to address performance limitations of LC/NC abstractions at scale, while visual components handle orchestration. Real-time insights into pipeline throughput, error rates, and system health are provided through observability platform integration, thereby facilitating continuous improvement and informed operational decision-making.

Table 2: Summary of LC/NC DevOps Framework Layers

Layer	Key Responsibilities	Enterprise Concerns Addressed
Interface	Visual drag-and-drop modeling with role-specific views and pre-built components.	Democratizes access, reduces onboarding time.
Orchestration	Translates models to workflows; integrates CI/CD, IaC, cloud APIs; template library.	Ensures consistency, supports multi-cloud.
Governance	Policy-as-code, RBAC, approvals, audit logs, shift-left security.	Prevents shadow IT, enforces compliance.
Execution	Runtime execution, resilience patterns, observability integration.	Handles scale, improves reliability.

4. Results and Discussion

The suggested framework has successfully addressed the major shortcomings of both traditional and Low Code / No Code (LC / NC) DevOps automation by enabling faster and more reliable delivery at an Enterprise level. The utilization of visual abstractions and reusable automation templates significantly decreases effort for development and pipeline configurations. Visual interfaces and standardized pipeline templates also reduce pipeline design time and the onboarding process for new teams and applications that support rapid digital initiative time-to-market. Standardized orchestration patterns provide consistent scalability across business units and geographic regions, thus eliminating the proportionate increase in operational staffing that is common in scripted DevOps implementations.

Compared to traditional DevOps methodologies, the proposed framework eliminates obstacles to participation from non-technical stakeholders contributing to delivery workflows. Technical complexities are abstracted using visual and model-driven components to enhance collaboration between business and IT teams while maintaining necessary operational control. Governance elements embedded within the framework, such as policy-driven controls and role-based access management, allow for regulatory compliance without compromising delivery velocity. Additionally, the inclusion of resilience patterns and observability tools aligns with existing DevOps best practices to improve deployment success rates and decrease mean time to recover in large-scale systems.

Although this framework has numerous advantages, there remain some challenges. The abstraction layers introduced by LC/NC platforms will result in performance overhead, particularly when operating in high concurrency or complex integration environments. However, hybrid approaches that utilize LC/NC platforms to manage orchestration and performance-critical components developed with custom code can mitigate performance concerns. Vendor lock-in remains a significant risk to enterprises that rely heavily on proprietary automation platforms. Therefore, open standards and multi-tool integrations are critical to avoid vendor lock-in risks. Finally, successful adoption of LC/NC DevOps will require mature governance and organizational alignment. Without dedicated platform teams to develop and maintain templates, enforce policies, and govern operations, LC/NC DevOps may be adopted in a fragmented manner across multiple business units. Therefore, although LC/NC DevOps automation is suitable for most enterprise delivery scenarios, it requires organizational maturity to realize long-term benefits.

5. Conclusion and Future Scope

The Low-Code / No-Code (LC/NC) DevOps Automation Model represents an important improvement over legacy enterprise software delivery models, improving upon issues related to scale, complexity, and reliance on specialized technical expertise. The authors present a conceptual Enterprise LC/NC DevOps Framework that consists of four primary layers: Interface, Orchestration, Governance, and Execution. By transforming low-level DevOps activities into high-level, visual, reusable, and policy-driven components, this framework will show how global enterprises can extend their DevOps processes across multiple distributed teams, while maintaining security compliance and operational consistency.

This approach aligns with prior research from both the enterprise DevOps and LC/NC spaces, highlighting the key roles that standardization, governance, and automation play in improving delivery performance and organizational agility. In addition to the technical advantages of this model, the authors emphasize the importance of embedding governance and security controls within the automation workflow rather than treating them as an external constraint, allowing enterprises to achieve an optimal balance between delivery velocity and risk management. As such, this framework provides practical recommendations for enterprises seeking to modernize their DevOps practices, reduce operational overhead, and minimize fragmentation across teams and platforms.

There are several possible paths for future research and technology development to investigate. Quantitative validation of the proposed framework via multi-case studies in large enterprises will provide evidence regarding the impact of the proposed framework on key performance metrics (e.g., deployment frequency, change failure rate, and compliance incidents). Advancements in Artificial Intelligence/Autonomous IoT Systems (AIOps) and other AI technologies offer additional possibilities for leveraging LC/NC DevOps Platforms through Predictive Failure Detection, Automated Pipeline Optimization, Intelligent Policy Enforcement, and Self-Healing Execution Workflows. Research could explore the integration of LC/NC DevOps Automation with Platform Engineering Initiatives, Cloud-Native Governance Frameworks, and Open Standards to mitigate vendor lock in and enhance interoperability.

Benchmarking of LC/NC DevOps automation relative to script-based DevOps methods in hybrid multi-cloud environments will provide a greater understanding of the long term performance, maintainability, and cost efficiency of each method. These areas of research collectively indicate that LC/NC DevOps automation has the potential to serve as both a technical innovation and a foundational engineering paradigm for sustainable digital transformation in global enterprises.

References

[1] A. van der Hoek et al., "Low-code software development: Opportunities and challenges," in *Proc. Int. Conf. Model-Driven Eng. Lang. Syst.*, 2022. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-16947-2_1

[2] A. Bosch et al., "Scaling DevOps in large organizations," in *Proc. ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3417990.3420204

[3] S. K. Dash et al., "Governance and security challenges in enterprise DevOps automation," *IEEE Access*, 2023.

[4] S. Shridhar, S. Shreyas, and S. Bose, "Analysis of low-code no-code development platforms in comparison with traditional development methodologies," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 9, no. 12, pp. 508–513, 2021. [Online]. Available: https://d1wqtxts1xzle7.cloudfront.net/78542364/Analysis_of_Low_Code_No_Code_Development_Platforms_in_comparison_with_Traditional.pdf

[5] A. Gasparini et al., "Low-code and no-code platforms: A systematic analysis of enterprise adoption challenges," *Complex Systems Informatics and Modeling Quarterly*, no. 36, pp. 52–67, 2023. [Online]. Available: https://csimq-journals.rtu.lv/csimq/article/view/csimq.2023-36.04

[6] M. Krief, "Learning DevOps: A comprehensive guide to accelerating software delivery," in *Proc. IEEE Int. Conf. Software Engineering Education and Training*, pp. 1–10, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/10163432

[7] N. P. Padhy et al., "Shift-left security: Integrating DevOps with secure development practices," *J. Artificial Intelligence, Machine Learning and Data Science*, vol. 5, no. 2, pp. 45–60, 2023. [Online]. Available: https://urfjournals.org/open-access/shifting-security-left-integrating-devops-with-secure-development-practices.pdf

[8] D. A. van der Burgh, "A readiness self-assessment model for low-code development enabled DevOps," M.Sc. thesis, Eindhoven Univ. of Technology, The Netherlands, May 2019. [Online]. Available: https://research.tue.nl/en/studentTheses/a-readiness-self-assessment-model-for-low-code-development-enable/

[9] S. Rafi, M. A. Akbar, M. Sánchez-Gordón, and R. Colomo-Palacios, "DevOps practitioners' perceptions of the low-code trend," in Proc. ACM/IEEE Int. Symposium on Empirical Software Engineering and Measurement, 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3544902.3546635

[10] M. Ccallo-Luque, A. Quispe-Quispe, C. Castillo-Sequera, and L. B. Ochoa, "An empirical study of developer discussions on low-code software development challenges," arXiv, Mar. 21, 2021. [Online]. Available: https://arxiv.org/abs/2103.11429.