myresearchgo Volume 2, Issue 5, May 2026, ISSN: 3107-3816 (Online)

Role of Low-Code Platforms in Modern Software Development

Author: Krupil Vekariya

Affiliation: University of Mumbai (IDOL) Email: krupilvekariya1954@gmail.com

Abstract

The rapid evolution of digital ecosystems has significantly increased the demand for agile, scalable, and cost-efficient software development methodologies. Traditional software engineering practices, while robust and flexible, often fail to meet modern requirements of speed and adaptability. Low-code development platforms have emerged as a disruptive innovation, enabling rapid application development through visual interfaces, model-driven architectures, and automated code generation.

This paper provides an in-depth analysis of low-code platforms, examining their architecture, operational mechanisms, and impact on the software development lifecycle (SDLC). It evaluates their effectiveness in enhancing productivity, reducing development costs, and democratizing software development through the rise of citizen developers. Furthermore, the study critically analyzes limitations such as scalability constraints, vendor dependency, governance challenges, and security risks.

Introduction

The field of software engineering has undergone significant transformation over the past few decades, evolving from structured programming to object-oriented design, agile methodologies, and DevOps practices. Despite these advancements, organizations continue to face challenges such as prolonged development cycles, increasing complexity of applications, and shortage of skilled developers.

In response to these challenges, low-code platforms have emerged as a novel approach that abstracts the complexities of programming. These platforms allow developers to build applications using graphical user interfaces, configuration-based logic, and reusable components rather than writing extensive lines of code.

Another important aspect is the emergence of citizen developers, who bridge the gap between business requirements and technical implementation. This shift represents a democratization of software development, where application creation is no longer limited to professional programmers.

Literature Review

The concept of low-code development originates from earlier methodologies such as Rapid Application Development (RAD) and model-driven engineering. These approaches emphasized faster development cycles and iterative design, which laid the foundation for modern low-code platforms.

However, existing literature also identifies several limitations. Scalability remains a concern for large-scale enterprise systems, and vendor lock-in poses risks related to long-term dependency. Security and compliance challenges are also frequently discussed, especially in industries dealing with sensitive data.

Overall, the literature suggests that while low-code platforms are highly effective for rapid development and prototyping, they must be used strategically within a broader software development framework.

1. Objectives of the Study

The primary objectives of this research are:

To understand the concept and architecture of low-code platforms To analyze their impact on modern software development processes To evaluate the advantages and limitations of low-code tools

To assess their role in improving productivity and reducing costs To explore their future potential in the IT industry

2. Research Problem

Modern organizations face several challenges in software development, including: Shortage of skilled developers

Increasing demand for faster application delivery High development and maintenance costs

Communication gaps between business stakeholders and technical teams

Traditional development approaches often struggle to address these challenges efficiently. While low-code platforms aim to provide solutions, they introduce new concerns such as limited customization, platform dependency, and potential security risks.

The central research problem of this study is to determine whether low-code platforms can effectively replace traditional development methods or whether they are better suited as a complementary approach.

3. Methodology

This study adopts a qualitative research methodology based on secondary data sources.

3.1 Data Collection

3.2 Data is collected from: Academic research papers

Industry reports (Gartner, Forrester)

Official documentation of low-code platforms

Case studies and real-world implementations

3.3 Platform Analysis

3.4 The study examines widely used low-code platforms such as: Microsoft Power Apps

OutSystems Mendix Appian

4. Evaluation Criteria

The platforms are evaluated based on:

Development speed Cost efficiency Ease of use Scalability

Security

4.1 Comparative Analysis

4.2 A comparison is conducted between low-code and traditional development approaches to understand their relative strengths and weaknesses.

5. Architecture and Key Features of Low-Code Platforms

Low-code platforms are built on model-driven and metadata-based architectures. They utilize abstraction layers that allow developers to define application logic visually, which is then converted into executable code by the platform.

Key Features:

Visual Development Interface

Enables application design using drag-and-drop tools Pre-built Components

Provides reusable templates and modules Integration Capabilities

Supports seamless integration with databases, APIs, and third-party services Cross-Platform Compatibility

Applications can run on web, mobile, and desktop environments Automation and Workflow Management

Built-in tools for business process automation

6. Advantages of Low-Code Platforms

6.1 Faster Development

Applications can be developed and deployed in significantly less time compared to traditional methods.

6.2 Cost Reduction

Reduced need for large development teams and shorter project timelines lower overall costs.

6.3 Accessibility

Non-technical users can participate in development, promoting inclusivity and innovation.

6.4 Flexibility

Applications can be easily modified and updated based on changing requirements.

6.5 Improved Collaboration

Enhances communication between business and IT teams, resulting in better alignment of goals.

7. Limitations of Low-Code Platforms

While low-code platforms offer significant advantages, their limitations are equally important for a balanced academic evaluation. These limitations are not merely technical constraints but also involve strategic, organizational, and long-term risks. A deeper understanding of these challenges is essential for effective adoption.

7.1 Limited Customization and Flexibility

Low-code platforms are designed around predefined components, templates, and workflows, which restrict flexibility.

Key Issues:

Difficulty in implementing complex business logic

Limited support for custom algorithms and advanced computations Constraints in modifying auto-generated code

For example, applications requiring:

Real-time analytics Advanced AI models

Complex financial calculations

often require traditional programming. Implication:

Low-code is best suited for standardized and repetitive applications, not highly customized systems.

7.2 Scalability Constraints

Scalability is one of the most critical limitations, especially for enterprise-level systems. Challenges:

Difficulty handling high user loads (concurrency)

Limited support for distributed systems and microservices architecture Performance degradation in data-intensive applications

Although platforms like OutSystems and Mendix are improving scalability, they may still not match the flexibility of custom-built architectures.

Implication:

Low-code platforms may not be ideal for: Banking transaction systems

Large-scale e-commerce platforms Real-time processing systems

7.3 Vendor Lock-In

Vendor lock-in is a significant strategic concern.

Causes:

Proprietary platform architecture

Platform-specific development languages and frameworks Limited portability of applications

Risks:

High cost of switching platforms Dependency on vendor updates and pricing Limited control over system evolution

Once an application is built on platforms like Microsoft Power Apps or Appian, migrating it to another platform can be complex and expensive.

Implication:

Organizations may face long-term dependency risks.

7.4 Security and Compliance Risks

Security is a major concern, particularly in regulated industries. Issues:

Limited visibility into backend security mechanisms Dependency on vendor for security updates

Challenges in meeting compliance standards (GDPR, HIPAA, etc.) Risks:

Data breaches Unauthorized access

Inadequate encryption or authentication mechanisms Implication:

Organizations must rely heavily on platform providers for security, which may not always align with internal policies.

7.5 Performance Limitations

Low-code platforms rely on abstraction layers, which can impact performance.

Challenges:

Slower execution compared to optimized custom code Inefficient handling of large datasets

Latency issues in real-time applications Performance bottlenecks are more evident in:

High-frequency transaction systems Real-time analytics platforms Implication:

Performance-critical applications may require traditional development.

8. Real-World Applications

Low-code platforms have moved beyond theoretical use and are now widely implemented across industries to solve real business problems. Their ability to rapidly develop, deploy, and modify applications makes them particularly valuable for process- driven, data-centric, and workflow-based systems.

Below is a detailed analysis of real-world applications across multiple sectors.

8.1 Banking and Financial Services

The banking sector is one of the largest adopters of low-code platforms due to the need for speed, compliance, and automation.

Key Applications:

Loan processing systems

Customer onboarding (KYC automation) Fraud detection dashboards

Internal risk management tools Example Use Case:

Banks use platforms like Appian to automate loan approval workflows: Customer submits application

8.2 Healthcare Industry

Healthcare organizations use low-code platforms to improve patient management and operational efficiency.

Key Applications:

Patient record management systems Appointment scheduling platforms Telemedicine applications

Hospital workflow automation Example Use Case:

Hospitals build patient management dashboards using Mendix: Centralized patient data

Real-time updates for doctors Automated alerts for critical cases

8.3 Retail and E-Commerce

Retail businesses require quick adaptation to market trends, making low-code ideal. Key Applications:

Inventory management systems Order tracking applications

Customer feedback systems Loyalty and rewards platforms

8.4 Enterprise Internal Applications

Organizations widely use low-code for internal tools that improve productivity. Key Applications:

HR management systems Employee onboarding portals Leave and attendance systems Internal communication tools Example Use Case:

Companies use Microsoft Power Apps to create:

Faster internal process automation Improved employee experience

8.5 Government and Public Sector

Governments use low-code platforms to improve digital services and citizen engagement.

Key Applications:

E-governance portals Public grievance systems

Online licensing and registration systems COVID-19 tracking dashboards

Example Use Case:

Government agencies develop citizen service portals: Online form submission

Automated processing Status tracking Benefits:

Faster service delivery Increased transparency Reduced paperwork

9. Future Scope

The future of low-code platforms is highly dynamic and closely aligned with broader technological advancements in software engineering. As organizations continue to prioritize speed, agility, and innovation,

low-code platforms are expected to evolve beyond simple application development tools into comprehensive ecosystems that support end-to-end digital transformation.

9.1 Integration with Artificial Intelligence and Machine Learning

One of the most significant future developments is the integration of Artificial Intelligence (AI) and Machine Learning (ML) into low-code platforms. These technologies will enhance automation and decision-making capabilities.

Future capabilities may include:

AI-assisted development: Systems that suggest UI designs, workflows, and logic automatically

Natural Language Processing (NLP): Converting user requirements written in plain language into functional applications

Predictive analytics integration: Embedding ML models directly into applications

This will reduce development effort further and move towards “no-code intelligent systems.”

9.2 Hyperautomation and Intelligent Workflows

Low-code platforms will play a central role in hyperautomation, which combines: Low-code development

Robotic Process Automation (RPA) Artificial Intelligence

Future systems will:

Automate complex business processes end-to-end Integrate human decision-making with automated workflows Reduce manual intervention across enterprise operations

This will significantly improve organizational efficiency and scalability.

9.3 Expansion of Citizen Development

The role of citizen developers is expected to grow substantially. Organizations will increasingly empower non-technical users to build applications.

Future trends include:

Formal training programs for citizen developers Development of governance frameworks to manage risks

Creation of collaborative environments where IT and business teams co-develop applications

However, organizations will need strong policies to prevent: Security vulnerabilities

Data mismanagement

Uncontrolled application development

9.4 Multi-Experience and Cross-Platform Development

Future low-code platforms will support multi-experience development, enabling applications to be built simultaneously for:

Web platforms Mobile devices Wearables

Internet of Things (IoT) devices This unified approach will:

Reduce development complexity

Ensure consistent user experiences across platforms Improve scalability of applications

9.5 Enhanced Scalability and Enterprise Adoption

Current limitations in scalability are expected to be addressed through: Cloud-native architectures

Microservices-based designs

Containerization technologies (e.g., Docker, Kubernetes)

Enterprise-grade platforms such as OutSystems and Mendix are already moving in this direction.

As a result:

Large enterprises will increasingly adopt low-code solutions

Complex applications will become more feasible within low-code environments

1. Conclusion

Low-code platforms have emerged as a transformative force in modern software development, fundamentally reshaping how applications are designed, developed, and deployed. By abstracting complex programming tasks into visual and model-driven processes, these platforms significantly reduce development time and lower

the technical barrier to entry. This shift enables organizations to respond more effectively to rapidly changing business environments, making low-code a critical enabler of digital transformation.

From an economic perspective, low-code platforms offer both direct and indirect cost benefits. Organizations can reduce reliance on large development teams, minimize maintenance overhead, and accelerate revenue generation through faster deployment.

Importantly, the analysis indicates that low-code platforms should not be viewed as a replacement for traditional software development, but rather as a complementary approach within a hybrid development model. While low-code excels in building data- driven applications, workflow automation systems, and internal enterprise tools, traditional coding remains essential for developing highly customized, performance-

intensive, and large-scale systems. Organizations that successfully integrate both approaches can achieve an optimal balance between speed and flexibility.

Looking ahead, the future of low-code platforms appears highly promising. The

integration of emerging technologies such as artificial intelligence, machine learning, and robotic process automation is expected to further enhance their capabilities.

In conclusion, low-code platforms represent a significant paradigm shift in software engineering, emphasizing speed, accessibility, and collaboration. Their successful adoption, however, depends on strategic implementation, proper governance, and a clear understanding of their strengths and limitations. When used appropriately, low- code platforms can significantly enhance organizational productivity, support digital transformation initiatives, and redefine the future of application development.

References

Books

Pressman, R. S. (2019). Software Engineering: A Practitioner’s Approach (9th ed.).

Missikoff, M. (2020). A simple methodology for model-driven business innovation and low code implementation. arXiv preprint arXiv:2010.11611.

Scharpf, D., Viljoen, A., Hein, A., & Krcmar, H. (2024). Business unit development: Benefits and challenges for employees. HMD Praxis der Wirtschaftsinformatik. □

Springer Industry Reports

Gartner. (2025). Magic Quadrant for Enterprise Low-Code Application Platforms.

Forrester Research. (2021). The Forrester Wave™: Low-Code Development Platforms for Professional Developers. □

Web Sources / Documentation

Microsoft Power Apps. (2024). Official documentation. Microsoft. Mendix. (2025). Platform overview and architecture. □

Mendix

OutSystems. (2025). Platform capabilities and architecture guide. □ OutSystems

Appian. (2024). Low-code automation platform guide.

L. C. Kasireddy, L. Popuri, G. Karunanithi, A. Varghese, S. Ahamad and Dharamvir, "Securing Business Data in Multi-Cloud Environments," 2025 International Conference on Digital Innovations for Sustainable Solutions (ICDISS), Faridabad, India, 2025, pp. 1-6, doi: 10.1109/ICDISS68238.2025.11320589.

L. C. Kasireddy, S. Paruchuri, C. Janakamma, A. Sarawat, K. C. Ravi and R. Kumar Chandu, "Cloud-Oriented IoT: Distributed Power-Aware Security Scheme with Data Integrity and Performance Enhancement," 2025 World Skills Conference on Universal Data Analytics and Sciences (WorldSUAS), Indore, India, 2025, pp. 1-6, doi: 10.1109/WorldSUAS66815.2025.11199185.

L. C. Kasireddy, A. Jeraldine Viji, P. K. Sholapurapu, D. Sowjanya Kolluru, D. U. Vishweshwar and P. Agrawal, "Intelligent Intrusion Detection using Artificial Bee Colony-Based Rule Discovery Techniques," 2025 IEEE Madhya Pradesh Section Conference (MPCON), Jabalpur, India, 2025, pp. 691-696, doi: 10.1109/MPCON66082.2025.11256592.

L. C. Kasireddy, S. Paruchuri, C. Janakamma, A. Sarawat, K. C. Ravi and R. Kumar Chandu, "Cloud-Oriented IoT: Distributed Power-Aware Security Scheme with Data Integrity and Performance Enhancement," 2025 World Skills Conference on Universal Data Analytics and Sciences (WorldSUAS), Indore, India, 2025, pp. 1-6, doi: 10.1109/WorldSUAS66815.2025.11199185.

J. L., L. Chandrakanth Kasireddy, R. V. Palanivel, G. Sushma, K. Bhimaavarapu and P. V. Reddy, "Predictive Modeling in Economics: The Role of AI and Deep Learning," 2025 World Skills Conference on Universal Data Analytics and Sciences (WorldSUAS), Indore, India, 2025, pp. 1-7, doi: 10.1109/WorldSUAS66815.2025.11199198.

N. Soni, L. C. Kasireddy, T. S., C. Sinhgadiya, S. Kumar and A. T. S., "A Recurrent Neural Network Framework for Effective DDoS Attack Detection in Cloud Computing," 2025 2nd International Conference on Multidisciplinary Research and Innovations in Engineering (MRIE), Gurugram, India, 2025, pp. 594-598, doi: 10.1109/MRIE66930.2025.11156616.

Jadhav, D., & Shinde, C. (2026). Sakhi: Stay safe stay fashionable. myresearchgo, 2(1), 1. https://doi.org/10.64448/myresearchgo.vol2.issue1.01.

Jadhav, A. (2026). AI-enhanced employee management system. myresearchgo, 2(1), 8. https://doi.org/10.64448/myresearchgo.vol2.issue1.02.

Rane, G., & Matteti, V. (2026). The evolution of the digital gaming ecosystem: A secondary analysis of PlayStation's market dominance and consumer retention strategies (2020–2026). Myresearchgo, 2(3), 1. https://doi.org/10.64448/myresearchgo.vol2.issue3.01.

Ansari, N., Sharma, A., & Yadav, S. (2026). The filtered classroom: AI-personalized learning and its implications for cultural exposure, empathy, and critical thinking. Myresearchgo, 2(3), 12. https://doi.org/10.64448/myresearchgo.vol2.issue3.02.

Junghare, P., Chheniya, J., Behare, M., Kashte, P., Belekar, S., Dhoble, V., & Kumari, S. (2026). Google's Neural Memory Architecture: A Comprehensive Review of the Titans Framework. Myresearchgo, 2(4), 75. https://doi.org/10.64448/myresearchgo.vol2.issue4.12.