PEER-REVIEWED JOURNAL OF COMPUTER SCIENCE (PRJCS)

ISSN Online : Open Access Scholarly Journal

Low-Code Development: When It Works and When To Code

Meena Jose Komban

Assistant Professor, Department of Computer Science, Yuvakshatra Institute of Management Studies (YIMS), Mundur, Kerala, India.

Article information

Received: 10th January 2026
Received in revised form:27th January 2026
Accepted: 3rd February 2026
Available online: 9th February 2026

Volume: 1
Issue: 2
DOI: https://doi.org/10.5281/zenodo.18898630

Abstract

Low-code development platforms have emerged as a significant force in enterprise software delivery, promising accelerated application development through visual interfaces and pre-built components. This paper evaluates the capabilities and limitations of low-code platforms through comparative analysis with traditional software development approaches. The study examines six application categories to determine where low-code platforms deliver genuine productivity gains and where traditional coding remains necessary. Drawing on market data, vendor assessments, and published case studies, the paper identifies internal workflow automation and departmental applications as the strongest use cases for low-code, while complex integrations, high-performance applications, and products requiring fine-grained customization continue to demand traditional development. A decision framework is proposed to help organizations determine the appropriate development approach for each project.

Keywords: - low-code development, no-code platforms, citizen development, rapid application development, software engineering

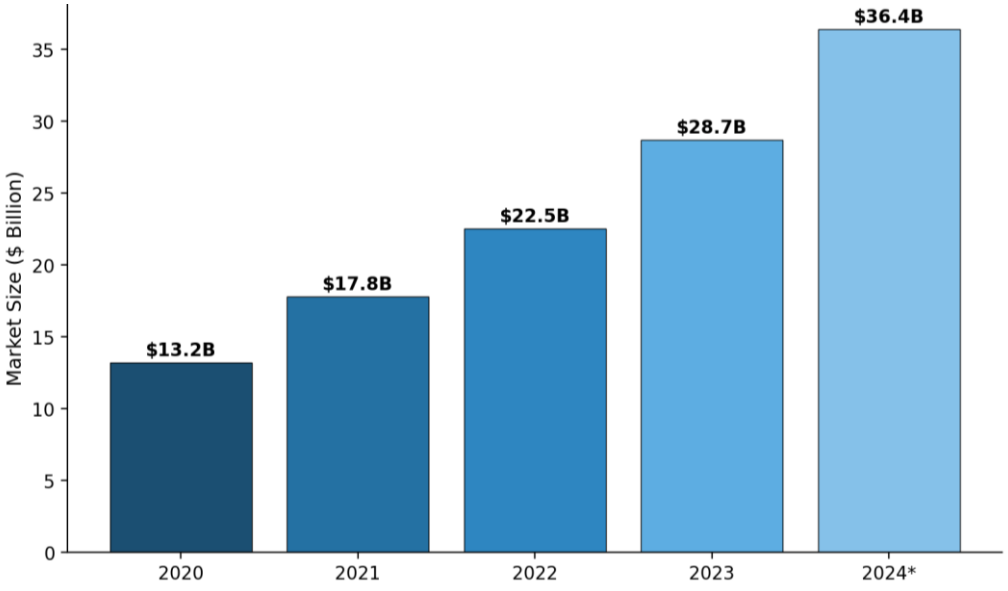
I. INTRODUCTION

The demand for custom software applications continues to outpace the supply of professional developers by a wide margin. Gartner estimates that enterprise IT demand for application development will grow at least five times faster than IT capacity to deliver it [1]. This gap has fueled the rapid growth of low-code and no-code development platforms, which enable users to create applications through graphical user interfaces, drag-and-drop components, and minimal hand-written code.

The global low-code development platform market has grown rapidly, with market estimates placing 2022 revenues above $20 billion and projections indicating continued strong growth [2]. Major vendors including Microsoft (Power Apps), Salesforce (Lightning), OutSystems, Mendix, and Appian have invested heavily in expanding platform capabilities, blurring the traditional boundary between professional development tools and business-user-oriented builders.

However, the enthusiasm surrounding low-code has generated inflated expectations. Not every application is a suitable candidate for low-code development, and organizations that adopt these platforms without a clear understanding of their limitations risk accumulating technical debt, vendor lock-in, and applications that cannot scale to meet evolving requirements [3]. This paper provides a balanced assessment to help IT teams make informed decisions about when to use low-code platforms and when traditional development is the more appropriate choice.

Figure 1: Global low-code development platform market size, 2020-2024 [2]. Market size figures are illustrative estimates based on published industry projections.



II. DEFINING LOW-CODE DEVELOPMENT

Low-code platforms provide visual development environments where applications are assembled primarily through configuration rather than programming. Forrester Research, which coined the term in 2014, defines low-code platforms as products that enable rapid application delivery with minimal hand-coding and quick setup and deployment [4].

These platforms typically offer visual data modeling, drag-and-drop interface builders, workflow automation engines, and pre-built connectors for common enterprise systems.No-code platforms represent a further abstraction, targeting business users with no programming experience.

While the distinction between low-code and no-code is often blurred, low-code platforms generally allow professional developers to extend applications with custom code, whereas no-code platforms restrict users to the capabilities provided by the visual builder [5]. This paper focuses primarily on low-code platforms that serve both professional developers and technically proficient business users.

III. STRENGTHS OF LOW-CODE PLATFORMS

A. Development Speed

The most cited advantage of low-code platforms is reduced development time. By providing pre-built UI components, data connectors, and deployment automation, these platforms eliminate much of the repetitive scaffolding work that consumes developer time in traditional projects.

Forrester reports that low-code platforms can substantially reduce development time for suitable application types, with vendor-reported reductions ranging from 50% to 90% [4]. This acceleration applies most strongly to CRUD (Create, Read, Update, Delete) applications, form-based workflows, and reporting dashboards.

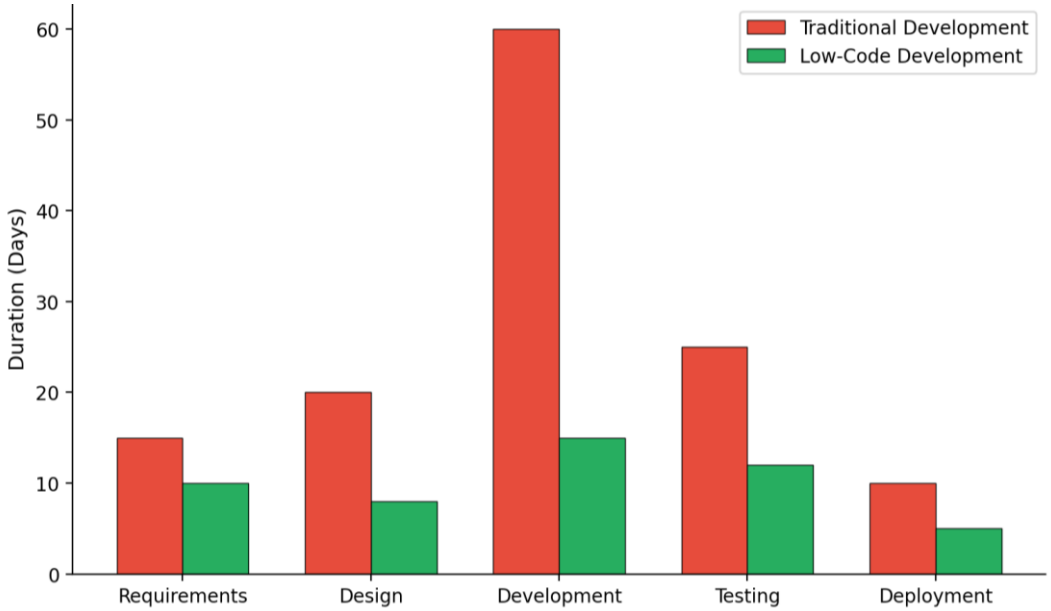
B. Citizen Development

Low-code platforms enable non-developers in business departments to build applications that address their own operational needs, reducing the backlog on central IT teams. This citizen development model works best when supported by governance frameworks that establish security standards, data access policies, and review processes [6]. Without governance, citizen development risks creating shadow IT assets that bypass security controls and data management policies.

C. Standardization

Applications built on low-code platforms inherit consistent design patterns, security controls, and deployment practices defined at the platform level. This standardization reduces the variability that often accompanies projects developed by different teams using different technology stacks [7].

Figure 2: Development lifecycle duration comparison: traditional vs. low-code [4]. Duration values are illustrative based on general industry benchmarks.



IV. LIMITATIONS AND RISKS

A. Customization Constraints

Low-code platforms impose boundaries defined by their visual builders and component libraries. Applications requiring highly customized user interfaces, complex business logic, or non-standard integrations frequently exceed these boundaries, forcing developers into workarounds that negate the platform's productivity benefits [8]. The more an application deviates from the platform's intended patterns, the more difficult it becomes to build and maintain.

B. Vendor Lock-In

Applications built on proprietary low-code platforms are tightly coupled to the vendor's infrastructure, data models, and runtime environment. Migrating a low-code application to a different platform or to traditional code typically requires a complete rewrite, as the visual models and platform-specific abstractions do not translate to portable formats [9]. This dependency gives vendors significant pricing leverage at renewal time.

C. Scalability Concerns

While low-code platforms handle moderate workloads adequately, applications that need to support thousands of concurrent users, process large data volumes, or meet strict performance requirements may encounter limitations. The abstraction layers that enable rapid development add runtime overhead that can impact response times and throughput under heavy load [10].

D. Security and Compliance

The ease of application creation on low-code platforms raises security concerns. Citizen developers may inadvertently expose sensitive data, create applications with insufficient access controls, or bypass data residency requirements. Organizations must establish governance frameworks that include security reviews, data classification policies, and deployment approvals for low-code applications [11].

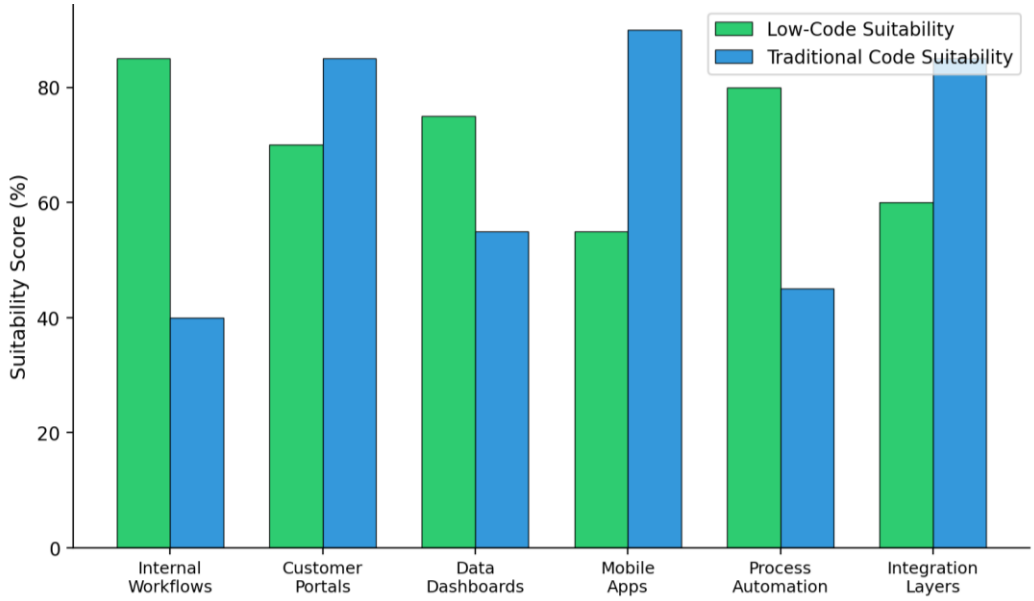
Table I. Low-Code Platforms: Strengths and Limitations

Dimension	Strength	Limitation
Development Speed	50-90% faster for suitable apps	Speed gain disappears for complex apps
Skill Requirement	Business users can build apps	Complex logic still needs developers
Maintenance	Platform handles updates	Version upgrades may break apps
Integration	Pre-built connectors available	Custom integrations are difficult
Scalability	Adequate for moderate loads	Performance ceiling under heavy load
Portability	N/A	Strong vendor lock-in

V. SUITABILITY ANALYSIS BY APPLICATION TYPE

The decision to use low-code versus traditional development should be driven by the specific characteristics of the application being built. Figure 3. compares the suitability of each approach across six common application categories, based on assessments from Gartner and Forrester [1][4].

Figure 3: Suitability comparison of low-code and traditional development by application type [1]. Suitability scores are illustrative assessments based on analyst reports.



Internal workflow applications, such as approval processes, expense management, and helpdesk ticketing, represent the strongest fit for low-code development. These applications have well-defined requirements, moderate complexity, and limited user bases. Conversely, customer-facing mobile applications that require pixel-perfect design, offline functionality, and platform-specific features remain better served by traditional development using native or cross-platform frameworks.

Table II. Decision Criteria for Low-Code vs. Traditional Development

Criterion	Favors Low-Code	Favors Traditional
User base	<500 internal users	Public-facing or high-concurrency
UI complexity	Standard forms/dashboards	Custom animations, complex UX
Integration needs	Standard APIs/databases	Legacy systems, custom protocols
Performance	Standard response times	Sub-100ms latency required
Lifespan	1-3 years	5+ years, long-term product
Team skills	Business analysts available	Full development team available

VI. GOVERNANCE FRAMEWORK FOR LOW-CODE ADOPTION

Organizations that adopt low-code platforms without governance structures frequently encounter application sprawl, data inconsistencies, and security gaps. A recommended governance framework includes a center of excellence that defines platform standards, conducts application reviews, and provides training for citizen developers [12]. Application classification tiers should determine the level of review required: simple departmental tools may need only automated security scans, while applications handling sensitive data or serving external users should undergo full security and architecture reviews.

VII. CONCLUSION

Low-code development platforms deliver genuine value for a specific category of enterprise applications, particularly internal tools, workflow automation, and departmental solutions with moderate complexity. The productivity gains for these use cases are well-documented and significant. However, low-code is not a replacement for traditional software development. Applications requiring deep customization, high performance, complex integrations, or long-term maintainability continue to demand the flexibility and control that only hand-written code provides. The most effective strategy treats low-code as a complement to traditional development, applying each approach where its strengths align

with project requirements. The decision framework presented in this paper equips IT teams to make this determination systematically rather than based on marketing claims or organizational pressure.

REFERENCES

[1] Gartner, "Magic Quadrant for Enterprise Low-Code Application Platforms," Gartner Research, Stamford, CT, USA, 2023.
[2] Statista, "Low-Code Development Platform Market Revenue Worldwide, 2020–2027," Statista Research Department, Hamburg, Germany, 2023.
[3] M. Rymer, J. Hammond, and R. Koplowitz, "The Forrester Wave: Low-Code Development Platforms for Professional Developers, Q2 2021," Forrester Research, Cambridge, MA, USA, 2021.
[4] C. Richardson and J. R. Rymer, "New Development Platforms Emerge for Customer-Facing Applications," Forrester Research, Cambridge, MA, USA, 2014.
[5] Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the Understanding and Comparison of Low-Code Development Platforms," in Proc. 46th Euromicro Conf. on Software Engineering and Advanced Applications, IEEE, 2020, pp. 171–178.
[6] Microsoft, "Power Platform Governance Best Practices," Microsoft Documentation, Redmond, WA, USA, 2023.
[7] OutSystems, "The State of Application Development 2023," OutSystems, Boston, MA, USA, 2023.
[8] N. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the Understanding and Comparison of Low-Code Development Platforms," in Proc. 46th Euromicro Conf. on Software Engineering and Advanced Applications, IEEE, 2020, pp. 171–178.
[9] G. Oppenlaender, J. Millimaggi, R. J. Morandi, and R. Schmidiger, "Mapping the Landscape of Low-Code Development Platforms: A Tertiary Study," in Proc. IEEE/ACM Int. Conf. Software Engineering Companion, IEEE, 2023, pp. 282–286.
[10] Mendix, "Performance Best Practices," Mendix Documentation, Rotterdam, Netherlands, 2023.
[11] OWASP, "Low-Code/No-Code Security Risks," OWASP Foundation, Wakefield, MA, USA, 2023.
[12] Gartner, "How to Establish a Center of Excellence for Low-Code Development," Gartner Research, Stamford, CT, USA, 2022.