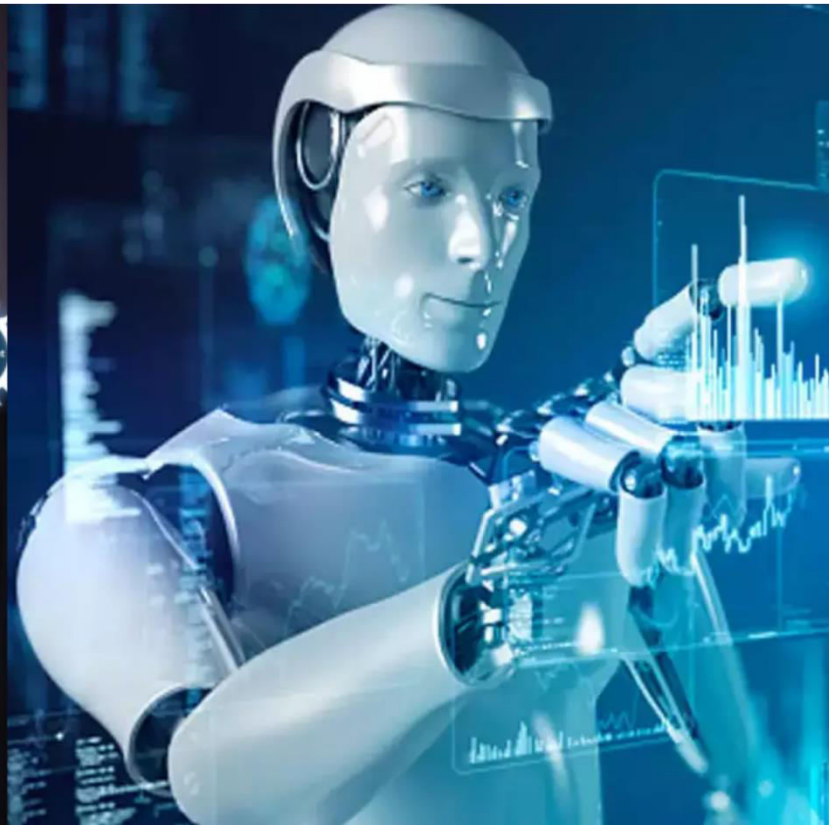
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 3, March 2025

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403118|

The Rise of Low-Code and No-Code Development Platforms: Impact on Traditional Software Development

Dr C K Gomathy, Mr. H.S.Vibhu Shreesh

Department of Computer Science and Engineering, SCSVMV University, Kancheepuram, Tamil Nadu, India

ABSTRACT: Low-code and no-code development platforms are rapidly gaining popularity, offering a simplified approach to software creation by enabling users with minimal technical skills to build functional applications. This paradigm shift is significantly impacting traditional software engineering practices. While these platforms promise faster development cycles, reduced costs, and greater accessibility, they also raise questions about scalability, security, and the evolving role of professional developers. This article explores the rise of low-code/no-code platforms, their advantages and limitations, and the ways in which they are reshaping the landscape of traditional software engineering. By examining both the opportunities and challenges, it provides insights into the future of software development in a world increasingly driven by automation and innovation

KEYWORDS: Accelerated Development Cycles - Democratization of Development - Focus on Business Logic - Customization and Scalability

I. INTRODUCTION

In today's fast-evolving digital world, businesses and organizations are increasingly relying on software to improve operations, enhance customer experiences, and drive innovation. Traditionally, developing software has been a time-consuming and resource-intensive process, requiring highly skilled engineers to write custom code and build complex systems from the ground up. However, the rise of low-code and no-code development platforms is reshaping this paradigm, making software development more accessible, faster, and cost-effective. Low-code and no-code platforms allow users, even those with minimal technical expertise, to create applications through intuitive, visual interfaces and drag-and-drop functionalities. Instead of writing code from scratch, users can configure and integrate pre-built components, significantly reducing development time and effort. As a result, businesses can respond to market changes and customer needs more quickly, while lowering the costs traditionally associated with software development. The growing adoption of these platforms is transforming the software engineering landscape in profound ways. While low-code/no-code tools are democratizing software development and empowering business users (often referred to as "citizen developers"), they are also raising questions about their impact on traditional software engineering. This article delves into the rise of low-code/no-code platforms, examining their challenges, and implications for the future and custom development.

II. PROBLEM STATEMENT

As businesses increasingly demand faster software solutions to adapt to rapidly changing market conditions, traditional software development processes often struggle to keep up with the speed, flexibility, and cost-efficiency required. This gap has led to the rise of low-code and no-code development platforms, which enable non-technical users to build applications with minimal coding effort. However, this shift brings challenges to traditional software engineering, including concerns around customization, scalability, security, and the evolving role of professional developers. Understanding the balance between the efficiency gains from these platforms and the complexities of maintaining robust, scalable, and secure software systems is crucial for organizations aiming to adopt low-code/no-code solutions effectively. This problem statement sets the foundation for exploring the benefits, challenges, and long-term impact of low-code/no-code platforms on the software engineering landscape.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403118|

III. IMPACT ON TRADITIONAL SOFTWARE DEVELOPMENT

1. Shifting Roles of Professional Developers

- From Coding to Architecture: As low-code/no-code platforms handle much of the repetitive coding work, professional developers are shifting their focus from writing basic code to higher-level tasks such as system architecture, integrations, and customizations. Developers are now more involved in ensuring that the underlying systems are scalable, secure, and adaptable.
- Advisory Roles: Developers are increasingly acting as advisors or "overseers" of low-code/no-code projects, helping citizen developers with more complex technical challenges, ensuring best practices are followed, and addressing issues related to performance or scalability.
- Custom Code Injection: In many cases, low-code platforms still require custom code for specific, complex features. Professional developers are tasked with creating these custom solutions within the platform's framework.

2. Changing Skill sets

- Focus on Integration and System Design: Traditional developers now need to be skilled in integrating low-code applications with other enterprise systems and databases. Understanding APIs, microservices, and cloud infrastructure has become critical as they work on making low-code applications part of a larger ecosystem.
- Collaboration and Communication: Since low-code/no-code platforms empower non-technical users to create applications, professional developers must now work more closely with business teams. This requires developers to improve their communication and collaboration skills to bridge the gap between technical and non-technical stakeholders.

3. Reduced Demand for Basic Coding

- Automation of Repetitive Tasks: Low-code/no-code platforms automate many routine tasks such as database CRUD operations, user interface design, and basic logic flows, reducing the need for developers to handle these activities manually.
- Fewer Entry-Level Coding Jobs: The demand for junior developers focused on basic coding might decrease as businesses leverage low-code/no-code platforms for simpler tasks. However, the need for specialized, high-level development skills remains strong for custom-built, enterprise-level applications.

4. Increased Focus on Quality Assurance and Security

- Ensuring Code Quality: As citizen developers build applications, the role of professional developers shifts toward ensuring that these apps meet quality standards. This includes reviewing code generated by the platform, performing performance tests, and ensuring that security practices are adhered to.
- Security Oversight: With more people building applications, including non-technical users, the risk of security vulnerabilities increases. Professional developers need to play a critical role in overseeing the security aspects of applications built on low-code/no-code platforms, ensuring they are compliant with regulatory requirements and free from common security flaws.

5. Impact on Complex and Large-Scale Applications

- Limitations of Low-Code/No-Code for Complex Projects: Low-code and no-code platforms are generally more suited to smaller, simpler applications. When it comes to building highly complex, large-scale systems (such as enterprise resource planning (ERP) or customer relationship management (CRM) systems), traditional software engineering is still essential. Professional developers are needed to create custom applications that handle complex business logic, data processing, and scalability challenges.
- Hybrid Development Approach: Many organizations are adopting a hybrid approach, using low-code/no-code platforms for front-end or simpler applications while relying on traditional development methods for back-end services, databases, and integration with core systems.

6. Rise of Hybrid Teams

- Collaboration between Developers and Citizen Developers: Low-code/no-code platforms encourage the formation of hybrid teams, where technical developers and non-developers collaborate on software projects.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403118|

Developers handle the more complex technical aspects, while business teams use no-code tools to address specific needs quickly.

- Blurring the Lines Between Business and IT: As more business users become involved in application development, the traditional divide between business teams and IT departments is becoming less pronounced. This collaborative environment allows for faster delivery of business solutions.

7. Technical Debt Concerns

- Risk of Accumulating Technical Debt: Since low-code/no-code platforms simplify the development process; applications can be built quickly without fully considering long-term maintenance and scalability. This can lead to the accumulation of technical debt, where applications require significant rewrites or updates as they grow or face increased complexity.
- Quality Control: Without the rigorous practices followed in traditional development (e.g., code review, testing), the quality of low-code applications might suffer, leading to future rework. Developers may find themselves fixing these issues later, adding to the long-term cost of ownership.

8. Evolution of Software Engineering Best Practices

- New Best Practices for Mixed Environments: As low-code/no-code platforms become more widespread; software engineering practices will need to evolve to integrate these tools effectively. This may involve new approaches to testing, documentation, version control, and application monitoring.
- Emphasis on Governance: IT departments will need to establish clear governance policies around the use of low-code/no-code platforms. This includes setting guidelines for when to use these platforms versus traditional development, how to manage data security, and how to ensure consistent code quality across applications.

9. Potential Job Shifts in Software Engineering

- Specialization in Platform Development: As more companies adopt low-code/no-code tools, there will be a demand for developers who specialize in extending these platforms, customizing them, and creating connectors or plugins for unique business needs.
- New Roles for Developers: Developers may transition into roles focused on managing and overseeing low-code platforms, ensuring that the solutions built align with enterprise standards and are integrated properly with core systems.

IV. CHALLENGES OF LOW-CODE AND NO-CODE PLATFORMS

1. Limited Customization and Flexibility

- **Pre-Built Components Restrictions:** While low-code/no-code platforms offer ready-made modules and templates, they can lack the flexibility needed for highly customized or complex applications. Businesses with specific requirements may find it difficult to implement features outside the platform's predefined options.
- **Complex Business Logic:** These platforms struggle with handling complex workflows, intricate business logic, or unique system architectures. This limitation often forces developers to fall back on traditional coding for certain aspects of a project.

2. Scalability Concerns

- **Handling Large-Scale Applications:** Low-code/no-code platforms are typically better suited for smaller, less complex applications. Scaling them to handle high volumes of data or a large user base can be challenging. As businesses grow, these platforms may struggle to meet performance expectations.
- **Performance Issues:** As applications built on these platforms grow in complexity, they may face performance bottlenecks due to the platform's underlying infrastructure, which is optimized for ease of use rather than efficiency or speed.

3. Security and Compliance Risks

- **Security Vulnerabilities:** Non-developers or citizen developers using these platforms might lack knowledge of security best practices, potentially introducing vulnerabilities into the applications. Additionally, the platforms themselves may have built-in security limitations that require careful oversight.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403118|

- **Regulatory Compliance:** In industries with strict regulatory requirements (e.g., healthcare, finance), low-code/no-code platforms may not offer the robust compliance features needed. Ensuring data privacy, encryption, and adherence to industry regulations can be more difficult with these platforms, especially for non-technical users.

4. Vendor Lock-In

- **Dependence on Platform Providers:** Businesses that rely heavily on a specific low-code/no-code platform risk becoming locked into the vendor's ecosystem. If the vendor changes pricing, limits support, or goes out of business, companies may face challenges migrating applications to another platform or solution.
- **Limited Control Over Platform Upgrades:** Companies relying on a particular platform are dependent on the vendor's roadmap for new features, bug fixes, and security patches. This can be problematic if the platform does not evolve in alignment with a company's needs.

5. Lack of Skilled Workforce for Customization

- **Gap in Advanced Development Skills:** While these platforms simplify development for basic applications, advanced customization still requires traditional software engineering skills. Finding developers who are well-versed in both low-code platforms and advanced coding can be challenging.
- **Reduced Focus on Core Engineering Skills:** Over-reliance on low-code/no-code platforms can lead to a diminished focus on core programming and engineering skills among developers. This may create a gap in talent when companies need to transition from platform-based development to custom-built solutions.

6. Limited Integration Capabilities

- **Challenges with Legacy Systems:** Integrating low-code/no-code applications with existing legacy systems or third-party services can be complex. Although some platforms offer integration options, they may not fully support custom integrations or may require additional development to connect to older systems.
- **Data Management Issues:** Data migration and synchronization between low-code applications and external systems can be problematic, particularly if the platform has limited database management capabilities or lacks support for complex data structures.

7. Collaboration Challenges between Developers and Non-Developers

- **Misalignment of Expectations:** When business users (citizen developers) and IT teams collaborate on low-code/no-code projects, there can be misalignments in terms of application expectations and technical feasibility. Business teams may push for rapid solutions without understanding the long-term impact on system performance or scalability.
- **IT Governance Issues:** Without proper oversight, citizen developers may build applications that bypass IT governance frameworks, potentially leading to compliance issues, security vulnerabilities, or conflicting software versions within the organization.

8. Platform-Specific Knowledge

- **Learning Curve for Advanced Features:** While low-code/no-code platforms are designed to be user-friendly, advanced customization often requires learning platform-specific concepts, making the learning curve steeper than initially expected. This can be particularly challenging for business users unfamiliar with the intricacies of software development.
- **Skill Transferability Issues:** Experience gained from working on a specific low-code/no-code platform may not transfer easily to other platforms or traditional development environments. Developers specializing in one platform might find it difficult to adapt their skills to other technologies.

V. FUTURE OF SOFTWARE ENGINEERING WITH LOW-CODE AND NO-CODE

1. Hybrid Development Approaches

Coexistence of Low-Code/No-Code with Traditional Development: Rather than replacing traditional software engineering, low-code/no-code platforms will complement it. These platforms will handle simpler, repetitive tasks and rapid prototyping, while complex, enterprise-level systems will still require traditional coding and engineering

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403118|

expertise. A hybrid development approach will emerge, where businesses use both models based on project requirements.

Selective Use Cases: Low-code/no-code platforms will become a go-to option for quick, internal applications, MVPs, and customer-facing apps that don't require heavy customization or complex logic. Traditional software engineering will focus on building custom applications, backend systems, high-performance applications, and systems that require heavy data processing or specialized features.

2. Increased Collaboration Between IT and Business

Citizen Developers Becoming Mainstream: Low-code/no-code platforms empower business users to take part in application development, blurring the lines between IT and business departments. This democratization of development will create hybrid teams where both professional developers and non-developers collaborate on software projects.

Faster Response to Business Needs With citizen developers directly addressing business challenges through these platforms, organizations can expect faster turnaround times for internal applications and quicker adaptation to market changes. IT teams will play a supportive role, ensuring security, scalability, and compliance while letting business teams innovate.

3. AI and Automation in Development

AI-Driven Development: The future of low-code/no-code platforms will increasingly incorporate artificial intelligence (AI) and machine learning (ML). AI-powered tools will assist in auto-generating code, suggesting features, and even performing code reviews. Automation tools will optimize the development process, reducing the manual effort required and making it even easier to build applications.

Automation for Testing and Deployment: Continuous integration and continuous deployment (CI/CD) pipelines will be integrated into low-code platforms, automating testing, deployment, and monitoring. This will make it easier to maintain quality and ensure that applications are delivered faster with fewer bugs.

4. Scalability and Complexity Management

Low-Code Platforms Evolving for Larger Applications: While low-code platforms are currently suited for smaller projects, the future will see them evolve to handle larger, more complex applications. Platform vendors are likely to introduce features that allow for better scalability, performance optimization, and management of complex business logic.

Modular Architectures: The adoption of modular architectures, where small, independent services (microservices) are combined to create more complex applications, will be essential. Low-code platforms will support this architecture, allowing developers to integrate pre-built modules with custom code for scalable applications.

VI. CONCLUSION

In conclusion, the rise of low-code/no-code development platformsmark a significant shift in the software engineering landscape, offering organizations faster development cycles, increased collaboration between business and IT, and empowering non-technical users to contribute to application creation. These platforms provide a solution for businesses seeking agility and cost-efficiency, particularly for less complex applications and rapid prototyping. However, their limitations, such as customization constraints, scalability issues, security concerns, and potential technical debt, mean that they are not a complete replacement for traditional software engineering. The future of software development will likely see a hybrid approach, where low-code/no-code platforms coexist with traditional methods. Professional developers will continue to play a vital role in managing complex systems, ensuring security, and overseeing the architecture of mission-critical applications. Ultimately, these platforms are reshaping the way software is built, but their success will depend on finding the right balance between speed and quality in the development process.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403118|

REFERENCES

1. C.K.Gomathy.(2010),"Cloud Computing: Business Management for Effective Service Oriented Architecture" International Journal of Power Control Signal and Computation (IJPCSC), Volume 1, Issue IV, Oct - Dec 2010, P.No:22-27, ISSN: 0976-268X .
2. C.K.Gomathy and Dr.S.Rajalakshmi.(2011), "Business Process Development In Service Oriented Architecture", International Journal of Research in Computer Application and Management (IJRCM) ,Volume 1,Issue IV, August 2011,P.No:50-53,ISSN : 2231-1009.
3. C.K.Gomathy and Dr.S.Rajalakshmi.(2012),"An Efficient Business Integration and Quality Service for Service Oriented Architecture", World Academy of informatics and management sciences(WAIMS), Volume 1,Issue II (WAIMS/0019/0084), March 2012,P.No: 74-79, ISSN : 2278-1315.
4. C.K.Gomathy and Dr.S.Rajalakshmi.(2012),"A proficient Business incorporation and Quality for SOA. ", International journal on information Science and computing, Volume 6,Issue II,July-2012, ISSN : 0973-9092.
5. C.K.Gomathy and Dr.S.Rajalakshmi.(2014),"A Business Intelligence Network Design for Service Oriented Architecture", International Journal of Engineering Trends and Technology (IJETT) ,Volume IX, Issue III, March 2014, P.No:151-154, ISSN:2231-5381.
6. C.K.Gomathy and Dr.S.Rajalakshmi.(2014),"Software Architecture Design Using Service Oriented on Quality Metrics", Australian Journal of Computer Science (AUJCS), Volume I,Issue I March 2014,P.No:09-16,ISSN:2251-3221.
7. C.K.Gomathy and Dr.S.Rajalakshmi.(2014),"Software Pattern Quality Comportment In Service Oriented Architectures", European Scientific Journal (ESJ) volume-10,No-9,Issue-March 2014,P.No-412-423,ISSN-1857-7881.
8. C.K.Gomathy and Dr.S.Rajalakshmi.(2014), "A Software Design Pattern for Bank Service Oriented Architecture", International Journal of Advanced Research in Computer Engineering and Technology(IJARCET), Volume 3,Issue IV, April 2014,P.No:1302-1306, ,ISSN:2278-1323.
9. C.K.Gomathy and Dr.S.Rajalakshmi.(2014),"A Software Ability link for Service Oriented Architecture", Global Journal of Management and Business Research (GJMBR), Volume(A) XIV, Issue-II, Version 1.0,May 2014, P.No:11-14, ISSN:2249-4588.
10. N.Sugumar, C.K.Gomathy, "Smart LCM for energy-efficient mechanism using CTX in wireless sensor networks" International Journal of Advance Foundation and Research in Computer (IJAFRC) Volume No.1 ,issue No. 4,IISSN-2348-4853.P.No 47-56
11. C.K.Gomathy and Dr.S.Rajalakshmi.(2014),"A Software Ability Network in Service Oriented Architecture", International Journal of science and Technology Education Research(IJSTER) , Volume 5,Issue II, June 2014,P.No:7-14, ISSN:2141-6559.
12. C.K.Gomathy and Dr.S.Rajalakshmi.(2014),"A Software Quality Metric Performance of Professional Management in Service Oriented Architecture", Proceedings of ICCTET'14, organized by Akshaya College of Engineering, Coimbatore. Archived in IEEE Xplore Digital Library, July 2014,ISBN:978-1-4799-7986-8.
13. C.K.Gomathy and Dr.S.Rajalakshmi.(2014), "Service Oriented Architecture to improve Quality of Software System in Public Sector Organization with Improved Progress Ability", Proceedings of ERCICA-2014, organized by Nitte Meenakshi Institute of Technology, Bangalore. Archived in Elsevier Xplore Digital Library, August 2014, ISBN:978-9-3510-7216-4.
14. C.K.Gomathy Dr.S.Rajalakshmi, M.Prema,"A framework for developing service-oriented architecture for mobile commuting-an exploratory study of SCSVMV university "Journal of Harmonized research in Engineering (JOHR) Volume No.2, Issue No.IV, October 2014,P.No:360-366, ,ISSN : 2347-7393
15. N.Sugumar, C.K.Gomathy," Energy-efficient mechanism using CTX In wireless sensor networks" Zenith International Journal of Multidisciplinary Research ,ISSN 2231-5780 Volume No.4 (5), MAY (2014), pp. 176-188.
16. C K Gomathy and V Geetha. Article: A Real Time Analysis of Service based using Mobile Phone Controlled Vehicle using DTMF for Accident Prevention. International Journal of Computer Applications 138(2):11-13, March 2016. Published by Foundation of Computer Science (FCS), NY, USA,ISSN No: 0975-8887
17. Dr.C K Gomathy and Dr.V.Geetha,Fake Job Forecast Using Data Mining Techniques, International Journal of Early Childhood Special Education (INT-JECSE)
18. Dr.C K Gomathy and Dr.V.Geetha, Multi-Source Medical Data Integration AndMining For Healthcare Services, International Journal of Early Childhood Special Education (INT-JECSE) DOI: DOI:10.9756/INTJECSE/V14I5.67ISSN: 1308-5581 Vol 14, Issue 05 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details