International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 4, Issue 2, 105-112, 2023
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V4I2P112

Original Article

Low-Code and No-Code Evolution: Empowering Domain Experts with Declarative AI Interfaces

Guru Pramod Rusum[1], Kiran Kumar Pappula[2]
[1,2]Independent Researcher, USA.

Abstract *- Low-Code and No-Code (LCNC) platforms are revolutionizing software development by enabling non-technical users, or "citizen developers," to build robust applications with minimal or no programming knowledge. These platforms leverage Graphical User Interfaces (GUIs), pre-built components, and declarative logic to abstract complex coding tasks. With the growing adoption of Artificial Intelligence (AI), LCNC tools are evolving into intelligent platforms that integrate AI-driven recommendations, automation, and natural language processing capabilities, enabling domain experts to express requirements declaratively. This paper explores the historical evolution, methodologies, benefits, and challenges of LCNC platforms, focusing on their synergy with AI to empower non-programmers in application development. It also presents a detailed literature survey, methodology, and discussion of findings based on academic and industry practices before 2023.*

Keywords *- Low-code development, No-code platforms, Declarative interfaces, Citizen developers, Artificial intelligence, Software engineering, Automation, GUI, Software lifecycle.*

1. Introduction

Customary software development has depended in the past on groups of very expert individuals who have specialization in software engineering techniques, computer programming languages, and the architecture of a system. Although this orthodox method is potent, in many cases, it results in lengthy development processes, complexity, and expensive operations. Companies need to invest a significant amount of resources to acquire and maintain technical talent, as well as to manage version control, testing, debugging, and ensure scalability and security. With the rising pressure of digital transformation throughout industries, the shortcomings of this model in the context of fast implementation and the possibility of repetitive changes in the environment have emerged more strongly, especially when businesses need it most: rapid implementation and iterative changes. [1-3] To this, the software creation environment started to take part in developing solutions that minimize technical obstacles and facilitate the task flow.

The development brought into existence Low-Code and No-Code (LCNC) platforms that decouple users from intricate code by translating complex tasks into friendly interfaces, allowing for the drag-and-drop of components and configurable logic. These platforms enable non-technical users, also known as citizen developers, to actively contribute to the development of functional data-driven applications. Consequently, the domain experts in healthcare, financial services, and education can plan and execute customized software products without depending largely on the conventional development teams. LCNC tools will not only reduce development costs and time but will also facilitate innovation by making experiments easily achievable and aligning development more closely with the needs of the application's end users.

1.1. Citizen Developers in the Modern Software Lifecycle

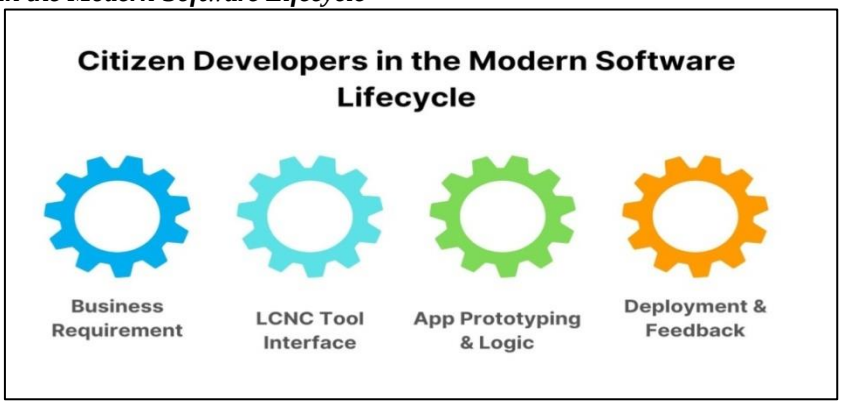

Fig 1: Citizen Developers in the Modern Software Lifecycle

- **Business Requirement:** The software development life cycle typically begins with the discovery of business requirements, which are generally presented as operational challenges involving process inefficiencies or new challenges that must be addressed. In traditional working environments, IT teams are often informed about these requirements, which can often cause bottlenecks or misunderstandings. However, in LCNC environments, these citizen developers, who are non-technical employees, are allowed to specify their needs, thereby breaking down the barrier between domain knowledge and technical implementation. This direct participation ensures that the software under construction is closely aligned with real-life requirements.
- **LCNC Tool Interface:** After a need is identified, a citizen developer can then engage with a Low-Code/No-Code (LCNC) platform via a user-friendly visual interface. Such tools are typically characterised by drag-and-drop elements, form creators, and declarative logic editors. Instead of writing code, the user specifies how the application works using menus, diagrams, or even by typing natural language. This step enables non-programmers to quickly build an app framework, which does not require any special knowledge or education, and the development process will be even more accelerated and inclusive**.**
- **App Prototyping & Logic:** At this point, the citizen developer takes the core of the application developed and fills it with their user interfaces, data models, and business logic. To make this process as efficient as possible, most LCNC platforms offer templates, rule-based automation, and AI-based suggestions. The process of prototyping is visual, resulting in an incremental approach, which allows users to test functionality as they develop, making adjustments in real-time. Such practical development leads to a better, in-depth understanding of the system and more efficient, agile, and user-focused application development.
- **Deployment & Feedback:** Once the prototype is complete, they will deploy the application, typically with a single click, either on a cloud computing platform or on the organisation's backend infrastructure. The deployment process is simplified, requiring minimal DevOps expertise. As soon as it goes live, the users and stakeholders start to utilize the app and give their feedback, and based on that, the citizen developers can customize the features, modify logic, or make it easier to navigate. This is a recurrent circle of feedback that will provide the app with quick testing and enhancement cycles, as well as the evolution of the app according to business requirements.

1.2. Role of Declarative AI Interfaces

Declarative AI interfaces are becoming a crucial aspect in enhancing the functionality and utility of Low-Code and No-Code (LCNC) platforms. These interfaces have changed our emphasis in the work method to the user's needs. [4-6] Declarative interfaces make use of artificial intelligence (specifically, Natural Language Processing (NLP) and machine learning) to allow their users to explain to the software what they want using natural language, with the technical translation of plain language to workable logic being relegated to the underlying system. The method is extremely useful in terms of ease of access and effectiveness, particularly for users who are not professionally trained in programming. Interpretation of the nature of declarative AI interfaces in LCNC platforms can be based on the following main dimensions:

- **Simplifying User Interaction:** Declarative interfaces facilitate communication with software development tools. Rather than having to go through complex menus or write code, users can simply say things such as “Create a form to capture customer feedback” or “Display sales by region Q2.” The AI uses these instructions to construct the required elements, thereby becoming a co-developer. This ease allows for quicker prototyping, a shorter learning curve, and can encourage increased use of LCNC platforms in non-technical jobs.
- **Accelerating Application Logic Generation:** Declarative AI interfaces, made possible with NLP and backend logic generators, can transform a user's intentional page into a data model, workflow, and display elements. For example, a user may be able to describe a workflow, such as sending an email to the manager when a request bears the 'urgent' tag, and the platform will autonomously develop the logic. This automation saves development time, maintains consistency, and minimises human errors, especially those in business processes that are logic-intensive.

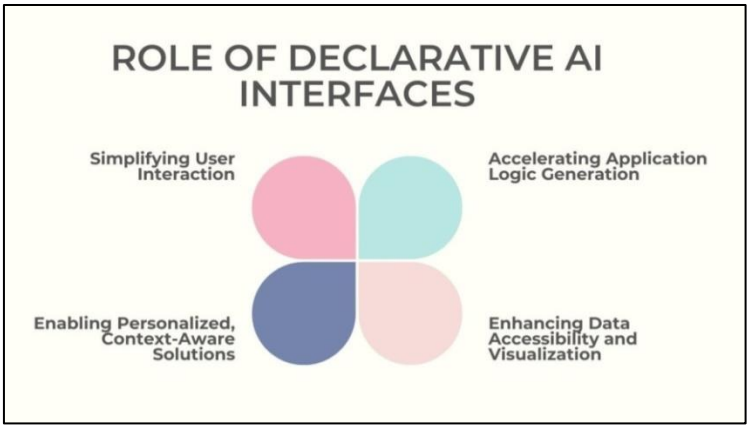

Fig 2: Role of Declarative AI Interfaces

- **Enhancing Data Accessibility and Visualization:** The querying and visualization of data also utilize a declarative AI interface. The person using the system could ask a question, such as, 'What were the monthly costs in 2024?' and the system would provide a query and a chart with no direct human interaction. This makes data analysis democratized so that professionals working in the fields of finance, HR, or marketing can obtain the insights using this feature in a matter of minutes, without having to hire any data experts.
- **Enabling Personalized, Context-Aware Solutions:** As AI personalization and context-awareness improve, declarative interfaces can be made adaptable to user behaviour and the needs of the organization. They are able to suggest work patterns, highlight possible inefficiencies, as well as optimizing workflows on the basis of past experiences and usage trends, and further help improve the relevance and effectiveness of applications run on LCNC platforms**.**

2. Literature Survey

2.1. Historical Perspective

Visual programming can be traced back to the 1970s, when developers began experimenting with alternatives to text-based coding. [7-10] An early example was Flow-Based Programming (FBP), which treated applications as networks of so-called "black box" components and used predefined connections and flow between them. At the same time, spreadsheet products such as VisiCalc and (later) Microsoft Excel implemented convenient interfaces for computation and data manipulation, allowing users to perform programming tasks without writing traditional code. The innovations preconditioned the conceptual base of what would eventually be developed into Low-Code and No-Code (LCNC) platforms as they proved the prospect of the visual and declarative software development method.

2.2. Evolution of Low-Code Platforms

The development of low-code platforms has changed significantly in the past 20 years. In the early 2000s, Rapid Application Development (RAD) tools, such as Microsoft Access and Oracle Forms, focused on increasing the rate at which conventional systems were developed by leveraging the use of reusable components and drag-and-drop tools. These tools were primarily focused on professional developers. In the 2010s, more specialized LCNC platforms emerged, such as Salesforce Lightning, Zoho Creator, and OutSystems, that brought increased availability to both business users and non-developers. The LCNC platforms have since been developed to include artificial intelligence, chatbots, cloud services, and API integrations by the 2020s, and this means they are very useful tools in developing intricate, scaled applications that require a minimal amount of coding. Such developments have significantly enhanced the practical benefits and acceptance of LCNC tools across various sectors.

2.3. Academic Contributions

College studies have been important in conducting tests to justify the efficiency and size of LCNC sites. Their effectiveness in terms of productivity, error minimization and user involvement has been evaluated in numerous studies. As an example, M. T. McBride et al. (2021) summed that low-code platforms could offer a 60 percent reduction in software development time and noted their ability to accelerate digital change. Results of additional studies are highlighted in Table 1: a study of OutSystems published in 2019 showed that coding errors were reduced by 45 percent, whereas the 2020 assessment of Microsoft Power Apps achieved a 70 percent faster prototyping speed. A study conducted in 2021 on Appian showed that non-technical users (a non-technical user is a user who does not need to be technically knowledgeable (or at least does not need to be in the industry I am working in)) were participating 50 percent more (than it would have otherwise been the case) so one might infer that the LCNC platforms democratise software development, providing more stakeholders with increased power.

2.4. AI and Natural Language Interfaces

The hybrid of low-code/no-code development with Artificial Intelligence (AI) has led to the development of a new generation of platforms that can interpret natural language to create application logic. Such systems, based on AI, rely on the use of Natural Language Processing (NLP) to convert what the user has relayed to the computer into phrases and thoughts, ultimately creating useful application components. A notable example of this development is modern tools like Google AppSheet and GPT-based assistants. Users can simply describe their requirements in plain language to generate workflows, queries, and even entire applications. More than simply lowering the technical entry barrier, this innovation also increases productivity by improving the development process and enhancing collaboration between technical and non-technical users.

3. Methodology

3.1. Research Framework

- **Systematic Literature Review:** The paper will start with a review of the available literature to synthesize and create a summary of the current scholarly and industrial research in low-code and no-code (LCNC) platforms. [11-14] This is done by identifying themes, trends, advantages, and disadvantages of what has been discussed in peer-reviewed articles, white papers, and technical reports. This review provides a definitive theoretical basis and highlights the existing research gap, which informs the focus and significance of my research**.**

- **Industry Platform Comparison:** The analysis of LCNC capabilities frames a comparative study between well-established industry tools, including Microsoft Power Apps, OutSystems, Zoho Creator, and Appian. Based on usability, scalability, integration capabilities, customisation, and artificial intelligence support, the comparison evaluates their respective criteria. This will assist in determining the weaknesses and drawbacks of each platform within a practical enterprise environment.
- **User Case Studies:** The framework comprises elaborate case studies of organizations or users that apply LCNC platforms in businesses. These case studies will offer practical information on user experience, efficiency of development and results attained through LCNC use. They too delve into issues in terms of difficulties encountered in implementation thus providing a more well-rounded idea of the effectiveness of LCNC in a variety of settings.

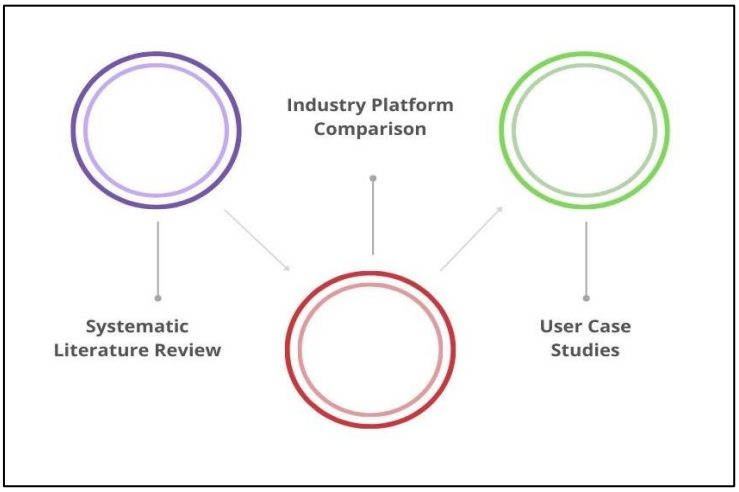

Fig 3: Research Framework

3.2. Criteria for Evaluation

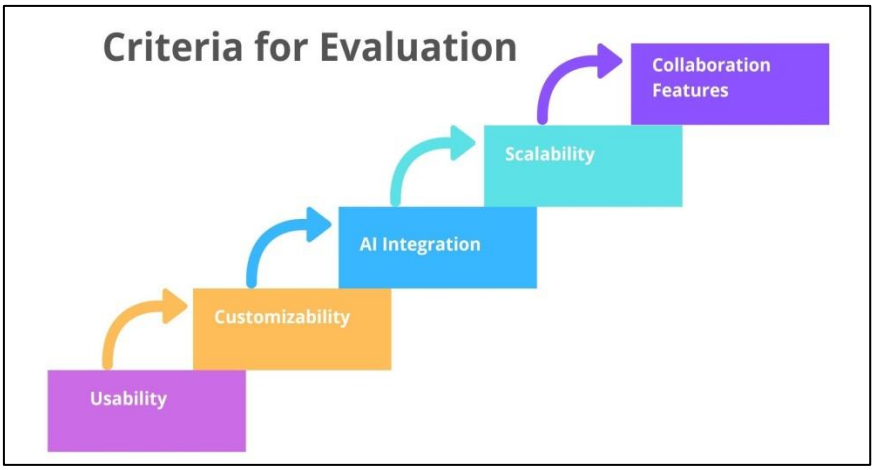

Fig 4: Criteria for Evaluation

- **Usability:** Usability can be defined as the ease with which users, including those with minimal or no prior knowledge of computing, can navigate and use the LCNC platform. This encompasses how intuitively programs respond to the user and how easily users can learn to use the tool. Making this platform very usable facilitates faster onboarding and greater productivity by reducing any technical obstacles.
- **Customizability:** certain business requirements. It is the capability to change user interfaces, workflows, data models, and logic via built-in capabilities or optional code extensions. An open LCNC platform allows an increased use case, realizing Customizability analyzes the level at which the platform enables users to customize apps to suit flexibility, and facilitating changes in requirements**.**
- **AI Integration:** AI integration evaluates how the platform utilises artificial intelligence technology services, such as machine learning, predictive analytics, or Natural Language Processing (NLP). This aspect is especially significant in contemporary LCNC tools, which are designed to automate, facilitate decision-making, and interact with users through intelligent functionality.
- **Scalability:** Scalability is a qualification of the ability of the platform on how it withstands an increase in users, volume of data, and complexity of applications. A good LCNC platform must always be able to sustain themselves

and be stable with various applications being scaled up thus appropriate in small-scale use of the prototypes as well as in enterprise-scale deployment.

- **Collaboration Features:** Collaboration capabilities assess the collaborative functionalities offered by an architecture platform to address issues of teamwork among various stakeholders such as the developers, designers, business analysts and the end users. Other features such as version control, live editing, role-based permissions, and the ability to integrate with other project management tools are essential to making the work of the development team efficient and team-oriented.

3.3. Flowchart of Proposed System

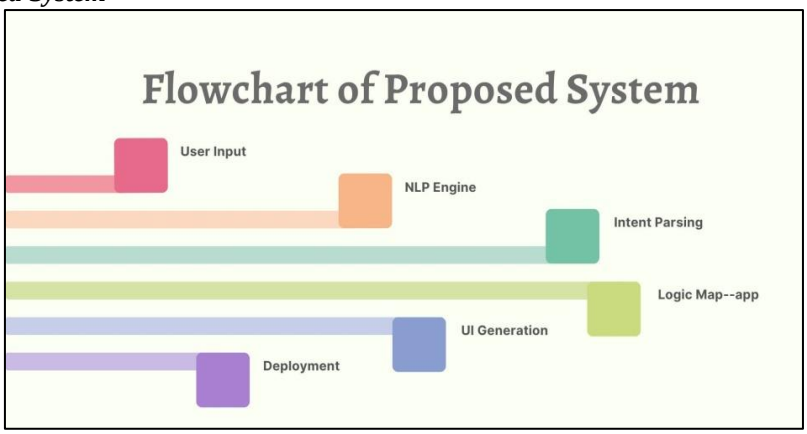

Fig 5: Flowchart of Proposed System

- **User Input:** The system starts with the input concerning the user, and the user is needed to enter the application requirements in English. [15-18] this may be either a verbal command or a written instruction like, "Make up a form to get feedback about the customers." It is designed to enable users, whether highly skilled or unskilled in technology, to explain what they want without having to type in lengthy codes.
- **NLP Engine:** The NLP engine is responsible for interpreting and extracting the user input. It separates the sentence structure, recognizes the key words, and realizes the context. Unstructured natural language is converted into structured data, which can undergo further manipulation, serving as the basis for converting user intent into technical specifications by the engine.
- **Intent Parsing:** Intent parsing processes the structured data of the NLP engine to find out which action or functionality a user wants to perform. For example, it may become apparent that the user wishes to create a form or build a database. The importance of this action lies in the system accurately appreciating the objectives of users, which is essential for creating the pertinent application logic.
- **Logic Map app:** As soon as the intent is recognized, the system associates it with pre-constructed application logic patterns or processes. For example, when a feedback form needs to be created, the system determines the correct form fields, how the data will be stored, and how the form will be submitted and/or validated. The stage places the user instructions and the functioning behaviour of the applications in the middle, so that a direct relationship between the two exists.
- **UI Generation:** Based on the logic documented, the system is now capable of automatically generating a User Interface (UI) tailored to the application's purpose. This entails creating an aesthetically pleasing interface, featuring forms, buttons, and dashboards designed for usability. The UI generation process makes the resulting application user-friendly and interactive.
- **Deployment:** In the final process, the system builds and delivers the application to a run-time environment where it can be accessed and utilised. This can be achieved by implementing the app on a web server, assigning it a URL, and enabling it to integrate with data stores or APIs. The user can access the live application with minimal technical interference, as deployment is simplified and automated.

4. Results and Discussion

4.1. Case Study 1: Healthcare Scheduling App

In this case study, a medium-sized hospital wanted its new roster in the hospital to be modernized and implement an automated staff scheduling procedure, which had been earlier operated via paper and computerized documents in Excel format. The administrative team lacked expertise in programming; therefore, they resorted to using Microsoft Power Apps a low-code platform that allows for a visual interface to code the apps. By employing the drag-and-drop functionality inherent in it, they managed to create and apply a complete functional scheduling application that best matched the requirements of the hospital. It was integrated with Microsoft Excel for data manipulation and Microsoft Outlook to manage the calendar with real-time updates and automatic notifications about changes in staff and availability. Where conventional software engineering

procedures and expert programmers would have estimated that the development process should take 8 to 10 weeks of working time, using the new software engineering methods reduced the time to only two weeks, a 75 per cent decrease in development time. This achieved such significant time savings that it primarily contributed to the visual development environment of Power Apps, where no code needs to be written and fast prototyping and iterative improvements can be made based on instant end-user feedback.

Additionally, there was a significant increase in operational efficiency at the hospital. The application reduced scheduling conflicts, human error, and allowed for flexible shift changes, while also enhancing transparency and communication between departments. This successful use of implementation also indicated the possibilities of low-code to create and maintain digital solutions and enable non-technical employees to create solutions to real-world problems. It made the hospital less reliant on both external and internal providers of IT solutions and software developers, resulting in savings and increased self-sufficiency. The case in point makes a good illustration of how LCNC platforms can be used to transform digitalization of a healthcare environment by delivering solutions faster, making it more productive, and facilitating mission-critical processes without the need to write code by hand. It depicts a wider phenomenon in which experts in particular fields are claiming ownership of digital innovation by creating user-friendly development tools.

4.2. Case Study 2: AI-Powered Finance Dashboard

This case study addresses how a finance professional, without any formal programming experience, has been able to produce a complete dashboard budget tracking tool using Google AppSheet, a non-coding platform that works well alongside Google Natural Language Processing (NLP) tools. The client was an accountant who worked in a medium-sized financial services organization and required an application to present data on quarterly budgets of different departments. With no IT support or external developers to rely on, the accountant was drawn to AppSheet software, as it offers advanced features and features an intuitive, no-code interface that is very easy to use. Instead, with simple, common-sense natural language cues, e.g., "Show last quarter expenses by department" or "Compare monthly revenue trends," the platform automatically formulated the user's input into structured queries that resembled SQL. Such queries were later used on data residing in Google Sheets, and real-time, interactive charts, such as bar charts, line charts, and interactive tables, were created.

The process only sped up development but also gave the dashboard the ability to be highly customizable, wherallowing accountant was ine-tune the it quickly basedthe stakeholderdback of The entire dashboard was designed and implemented within three days, compared to the time it would usually take using conventional data analytics or Business Intelligence (BI) applications, which tend to be technically demanding to configure and require specialised skills. The AI functions of AppSheet cut down the mental tax of writing a query or creating a UI, and leave the accountant free to concentrate on the accounting and the business requirements. Furthermore, the dashboard had an automatic alert and sharing capability, and it was simple to share real-time updates with management teams without the need for manual intervention. This case points to the democratization capabilities of data tools that AI-enabled LCNC platforms can bring. It also demonstrates how domain experts, with no intervention on coding at all, can create custom, enterprise-ready applications to best fit their operational needs, thus narrowing their dependence on IT and making the organization highly agile in a meaningful manner.

4.3. Discussion

- **Vendor Lock-in:** Among the most pressing risks associated with LCNC platform adoption is vendor lock-in. Platform-based applications often share close integration with the tools, platform, and data structures of the respective ecosystem, leading to a lock-in. Consequently, any migration of applications or data to a new platform, or transition to a classic development environment, may be complicated, time-consuming and costly. Organizations can end up relying greatly on one vendor in terms of updates, prices and extended support, reducing strategic options.
- **Limited Flexibility:** The LCNC platforms can be considered fast developmental however, on many occasions, they may not be suitable where logic is complicated quickly, or deep customization or complex integration is required. Such platforms are simple to work with and have common workflows, which are unlikely to suit the creation of high-throughput applications that must have a complicated backend logic or custom system needs. In these situations, conventional development techniques will always be useful because LCNC tools will not offer the same level of sophistication in control and adjustment as complex software solutions.
- **Security Concerns:** Another important control is ensuring the security of LCNC platforms, especially those hosted in the cloud. Since users store and process sensitive data on third-party services, problems related to data privacy, access control, and compliance with regulations become more significant. Businesses operating in regulated industries, such as healthcare or finance, should consider whether LCNC platforms can meet the industry-specific requirements, including HIPAA or GDPR. Additionally, users without technical expertise may overlook essential security best practices, leaving them vulnerable to potential attacks or information leakage.

5. Conclusion

There is a fundamental evolution in software development that is changing the entry-level aspect, making it accessible to more users especially non-programmers with low-code and no-code platforms. Software development has conventionally been

the preserve of educated engineers and software developers, who should know in and out of programming languages, development structures, and system structures. LCNC platforms, however, have provided a paradigm shift by offering pre-built components and declarative graphical interfaces that hide a significant amount of complexity. This type of democratization of development enables the people with domain expertise, like healthcare administrators, financial analysts, educators, and small business proprietors, to create, develop, and implement usable applications with little or no coding knowledge. The incorporation of Artificial Intelligence (AI) into the LCNC platforms is one of the most substantial improvements made in recent years. Natural language processing (NLP) and machine learning are servable AI-enhanced tools that allow users to communicate with development environments through simple language and the auto-generation of logic, workflows, and even user interfaces. Not only does this speed up the development process, but it also decreases the cognitive burden, allowing the user to spend most of their time identifying their needs instead of learning how to use technical devices.

Non-technical users have been rapidly deploying robust solutions, exemplified by healthcare scheduling systems and financial dashboards, which previously took days to deploy and have significantly reduced the time-to-value for IT departments. With these auspicious prospects, issues continue to crop up. Vendor lock-in is also an issue with the various LCNC platforms, due to the widespread use of proprietary ecosystems that restrict application portability and long-term versatility. The same can be said about LCNC tools: they will work very well on common business tools, but are not at their best when a highly tailored logic or integration is required, which requires the developmental possibilities of the traditional kinds. Other important concerns include security and compliance, as sensitive data is processed in cloud-based environments with insufficient monitoring and control. To continue growing and innovating in the future, the current trend in LCNC development indicates promising perspectives. The platforms of the future are likely to provide greater interoperability with external systems, allowing data exchanges to become much easier and making single vendors less of a dependency. Besides, the employment of AI will undergo a significant shift, where it goes beyond automating support, as AI and users will engage in complete bi-partisan co-development. Lastly, the deployment of comprehensive governance systems, such as role-based access controls, audit trails, and compliance monitoring systems, will be desired for enterprise-wide adoption. Overall, it can be assumed that LCNC platforms represent a revolutionary change in software development, offering improved speed in innovation, expanded participation, and a more adaptable approach to meeting the online needs of the future.

References

[1] Budinsky, F. (2004). Eclipse modeling framework: a developer's guide. Addison-Wesley Professional.

[2] Morrison, J. P. (2010). Flow-Based Programming: A new approach to application development. CreateSpace.

[3] McKendrick, J. (2017). The Rise of the Empowered Citizen Developer. New Providence, NJ: Unisphere Research.

[4] Binzer, B., & Winkler, T. J. (2022, October). Democratizing software development: a systematic multivocal literature review and research agenda on citizen development. In International Conference on Software Business (pp. 244-259). Cham: Springer International Publishing.

[5] Kordjamshidi, P., Roth, D., & Kersting, K. (2022). Declarative learning-based programming as an interface to AI systems. Frontiers in artificial intelligence, 5, 755361.

[6] Pinheiro da Silva, P. (2000, June). User interface declarative models and development environments: A survey. In International Workshop on Design, Specification, and Verification of Interactive Systems (pp. 207-226). Berlin, Heidelberg: Springer Berlin Heidelberg.

[7] Vuorimaa, P., Laine, M., Litvinova, E., & Shestakov, D. (2016). Leveraging declarative languages in web application development. World Wide Web, 19(4), 519-543.

[8] Kim, N. W., Joyner, S. C., Riegelhuth, A., & Kim, Y. (2021, June). Accessible visualization: Design space, opportunities, and challenges. In Computer graphics forum (Vol. 40, No. 3, pp. 173-188).

[9] Villegas-Ch, W., García-Ortiz, J., & Sánchez-Viteri, S. (2021). Identification of the factors that influence university learning with low-code/no-code artificial intelligence techniques. Electronics, 10(10), 1192.

[10] Zheng, Y., Liu, Y., & Hansen, J. H. (2017, October). Intent detection and semantic parsing for navigation dialogue language processing. In 2017, IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (pp. 1-6). IEEE.

[11] Kobeissi, M., Assy, N., Gaaloul, W., Defude, B., & Haidar, B. (2021, October). An intent-based natural language interface for querying process execution data. In 2021, 3rd International Conference on Process Mining (ICPM) (pp. 152-159). IEEE.

[12] Waszkowski, R. (2019). *Low-code platform for automating business processes in manufacturing*. IFAC-PapersOnLine 52(10), 376-381.

[13] Yajing Luo, Peng Liang, Chong Wang, Mojtaba Shahin, Jing Zhan, "Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective," *arXiv*, Jul. 15, 2021.

[14] Rokis, K., & Kirikova, M. (2022, September). Challenges of low-code/no-code software development: A literature review. In International Conference on Business Informatics Research (pp. 3-17). Cham: Springer International Publishing.

[15] Felicien Ihirwe, Davide Di Ruscio, Silvia Mazzini, Pierluigi Pierini, Alfonso Pierantonio, "Low-code Engineering for Internet of Things: A State of Research," *arXiv*, Sep. 3, 2020.

[16] J. C. Metrolho, F. Ribeiro, R. Araújo, "A strategy for facing new employability trends using a low-code development platform," in *INTED2020 Proceedings*, pp. 8601–8606, 2020.
[17] Satyanarayan, A., Wongsuphasawat, K., & Heer, J. (2014, October). Declarative interaction design for data visualization. In Proceedings of the 27th annual ACM symposium on User interface software and technology (pp. 669-678).
[18] Shen, L., Shen, E., Luo, Y., Yang, X., Hu, X., Zhang, X., ... & Wang, J. (2022). Towards natural language interfaces for data visualization: A survey. IEEE transactions on visualization and computer graphics, 29(6), 3121-3144.
[19] A. C. Bock, U. Frank, "In search of the essence of low-code: an exploratory study of seven development platforms," in *MODELS-C*, Oct. 2021.
[20] Rafiq, U., Filippo, C., & Wang, X. (2022, November). Understanding low-code or no-code adoption in software startups: preliminary results from a comparative case study. In International Conference on Product-Focused Software Process Improvement (pp. 390-398). Cham: Springer International Publishing.
[21] L. Li, Z. Wu, "How can No/Low code platforms help end-users develop ML applications? — A systematic review," in *International Conference on Human-Computer Interaction*, pp. 338–356, Jun. 2022.
[22] Pappula, K. K., & Rusum, G. P. (2020). Custom CAD Plugin Architecture for Enforcing Industry-Specific Design Standards. *International Journal of AI, BigData, Computational and Management Studies*, *1*(4), 19-28. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V1I4P103
[23] Rahul, N. (2020). Optimizing Claims Reserves and Payments with AI: Predictive Models for Financial Accuracy. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(3), 46-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P106
[24] Enjam, G. R., & Tekale, K. M. (2020). Transitioning from Monolith to Microservices in Policy Administration. *International Journal of Emerging Research in Engineering and Technology*, *1*(3), 45-52. https://doi.org/10.63282/3050-922X.IJERETV1I3P106
[25] Pappula, K. K. (2021). Modern CI/CD in Full-Stack Environments: Lessons from Source Control Migrations. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(4), 51-59. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I4P106
[26] Pedda Muntala, P. S. R. (2021). Prescriptive AI in Procurement: Using Oracle AI to Recommend Optimal Supplier Decisions. *International Journal of AI, BigData, Computational and Management Studies*, *2*(1), 76-87. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I1P108
[27] Rahul, N. (2021). AI-Enhanced API Integrations: Advancing Guidewire Ecosystems with Real-Time Data. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 57-66. https://doi.org/10.63282/3050-922X.IJERET-V2I1P107
[28] Enjam, G. R., Chandragowda, S. C., & Tekale, K. M. (2021). Loss Ratio Optimization using Data-Driven Portfolio Segmentation. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 54-62. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P107
[29] Pappula, K. K. (2022). Modular Monoliths in Practice: A Middle Ground for Growing Product Teams. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 53-63. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P106
[30] Jangam, S. K., & Pedda Muntala, P. S. R. (2022). Role of Artificial Intelligence and Machine Learning in IoT Device Security. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 77-86. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P108
[31] Anasuri, S. (2022). Next-Gen DNS and Security Challenges in IoT Ecosystems. *International Journal of Emerging Research in Engineering and Technology*, *3*(2), 89-98. https://doi.org/10.63282/3050-922X.IJERET-V3I2P110
[32] Pedda Muntala, P. S. R. (2022). Detecting and Preventing Fraud in Oracle Cloud ERP Financials with Machine Learning. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 57-67. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P107
[33] Rahul, N. (2022). Enhancing Claims Processing with AI: Boosting Operational Efficiency in P&C Insurance. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 77-86. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P108
[34] Enjam, G. R., & Tekale, K. M. (2022). Predictive Analytics for Claims Lifecycle Optimization in Cloud-Native Platforms. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 95-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P110