UDC:

DOI: 10.5937/AnEkSub2500007R

Review article

Анали Економског факултета у Суботици – The Annals of the Faculty of Economics in Subotica
Vol. XX, No. XX, pp. XX-XXX
Received: 05/03/2025
Revision: 05/05/2025
Accepted: 15/05/2025
Published online: 20/05/2025

Low Code/No Code Software Development: Definition, Drivers, and Inhibitors

Low-Code/No-Code razvoj softvera: definicija, pokretači i inhibitori

Lazar Raković
Univeristy of Novi Sad, Faculty of Economics in Subotica, Subotica, Serbia
lazar.rakovic@ef.uns.ac.rs https://orcid.org/0000-0002-1465-588X
Lena Đorđević Milutinović
University of Belgrade, Faculty of Organizational Sciences, Belgrade, Serbia
lena.djordjevic.milutinovic@fon.bg.ac.rs https://orcid.org/0000-0002-3758-0147
Paweł Lula
Krakow University of Economics, Kraków, Poland
lulap@uek.krakow.pl https://orcid.org/0000-0003-2057-7299
Vuk Vuković*
Univeristy of Novi Sad, Faculty of Economics in Subotica, Subotica, Serbia
vuk.vukovic@ef.uns.ac.rs
Marek Dziura
Krakow University of Economics, Krakow, Poland
dziuram@uek.krakow.pl https://orcid.org/0000-0002-4889-2883
Tomasz Rojek
Krakow University of Economics, Krakow, Poland
rojekt@uek.krakow.pl https://orcid.org/0000-0002-2977-4312

Abstract

Purpose: Low Code and No Code refer to software development by end users with little or no IT background. The goal of the manuscript is to define this concept, as well as identify the drivers on one hand, and the limitations, challenges, and inhibitors on the other.
Methodology: In order to answer the previously posed research questions, a systematic literature review was conducted. The systematic review of the literature included three main steps: planning the review, conducting the review, and writing the report.
Findings: The paper presents various definitions of the LowCode/NoCode concept, as well as one general definition. It then lists the key drivers that lead to the increasing use of this concept in organizations. After that, the limitations, challenges, and inhibitors of Low Code/No Code software development are presented.
Originality/value: The paper provides a clear and systematic review of the LowCode/NoCode concepts in relation

* Corresponding author

to three dimensions (definition, drivers, and limitations/challenges/inhibitors).
Practical implications - The results presented in the paper can be useful for both IT departments and business units as a starting point for establishing and managing LowCode/NoCode practices within an organization.
Limitations: The systematic literature review included papers published in two citation databases and in English. Future research directions would focus on the empirical validation of specific drivers and limitations, challenges, and inhibitors from both business and IT users' perspectives.

Keywords: Low Code, No Code, Development, Citizen Development
JEL classification: **M10, Z00**

Сажетак

Циљ: Low Code and No Code представљају приступ развоју софтвера од стране крајњих корисника који нису ИТ стручњаци. Циљ рада је да дефинише овај концепт, као и да идентификује покретаче с једне стране и ограничења, изазове и препреке с друге.
Методологија Како би се одговорило на претходно постављена истраживачка питања, спроведен је систематски преглед литературе. Систематски преглед литературе обухвата три главна корака: планирање прегледа, спровођење прегледа и писање извештаја.
Резултати: Рад представља различите дефиниције концепта Low Code/No Code, као и једну општу дефиницију. Затим се наводе кључни покретачи који доводе до све веће употребе овог концепта у организацијама. Након тога, представљена су ограничења, изазови и препреке Low CodeNo Code софтверског развоја.
Оригиналност/вредност – Рад пружа јасан и систематичан преглед концепата Low Code/No Code у односу на три димензије: дефиниција, покретачи и ограничења/изазови/препреке.
Практична примена – Резултати представљени у раду могу бити корисни како за ИТ одељења, тако и за пословне јединице као полазна тачка за успостављање и управљање Low Code/No Code праксама у оквиру организације.
Ограничења истраживања: Систематски преглед литературе обухватио је само радове објављене у две цитатне базе података и на енглеском језику. Будући правци истраживања били би усмерени на емпиријску валидацију одређених покретача и ограничења, изазова и препрека из перспективе како пословних, тако и ИТ корисника.

Кључне речи: Low Code, No Code, развој софтвера, развој софтвера од стране крајњих корисника.
ЈЕЛ класификација: M10, Z00

Introduction

The digital transformation of organizations is an inevitable process that has affected almost every organization, regardless of the activity it performs (Modupe, 2023; Popović, 2020; Ubiparipović et al., 2023). Consequently, organizations are in need of more software solutions that automate business processes (Binzer et al., 2024; Elshan et al., 2024; Prinz et al., 2024). It appears, however, that the number of required IT developers does not match the number of available ones (Biedova et al., 2024; Guthardt et al., 2024; Käss et al., 2023a, 2023b). In response, large software companies are increasingly promoting software development tools that end users without adequate IT skills can use (Sahay et al., 2020). Literature and practice refer to these platforms as Low Code/No Code (LCNC) platforms, and software as *Low Code/No Code development* (LCNCD). This concept is becoming increasingly popular in both academia and within the software industry (Pinho et al., 2023), as well as in non-IT organizations (Guthardt et al., 2024). The term low code/no code has become a new jargon in the software community (Rafiq et al., 2022) and is gaining importance and attracting the attention of many organizations (Luo et al., 2021; Overeem & Jansen, 2021). Although Low Code/No Code tools are capable of accelerating the digital transformation (Martinez et al., 2024), according to Kass et al. (2022) research in this area has just begun, and further research is needed.

In spite of the lack of a formal and clear definition of Low Code/No Code (LCNC) (Luo et al., 2021; Pinho et al., 2023), it is true that this type of software development has emerged as a relatively new, increasingly important, rapidly improving area of software development by end-user developers, with no manual coding necessary, making it easier to create business applications, and bypassing traditional IT bottlenecks (Beranic et al., 2020; Biedova et al., 2024; De Silva et al., 2024; Luo et al., 2021; Pinho et al., 2023; Rokis & Kirikova, 2022). This concept emerged in the early 2000s when drag-and-drop tools for web page development became available (Biedova et al., 2024). Furthermore, it is impossible to ignore the concepts that preceded and contributed to LCNC development. In their literature review, Schenkenfelder and colleagues (2024) report that end user development (EUD) was introduced in the 1980s, while consulting firms coined the term Low Code development in 2014. The same authors state that LC platforms or services are usually cloud-based and focused on business processes, user interfaces, databases, web and mobile applications, and industrial applications. Di Ruscio et al. (2022) compare model driven engineering with low code development and point out that both approaches have similar goals, but with some differences, since not all model driven engineering techniques are aimed at reducing the amount of code, nor are all low-code approaches driven by models. According to the same authors, over the past few decades, several trends have been distinguished that attempted to reduce manual code writing, including 4GL and CASE tools in the 1980s, Rapid Application Development in the 1990s, End User Development a decade later, and Model Driven Development in the last two decades. There is also Shadow IT (Đorđević Milutinović et al., 2023; Raković, 2020; Raković et al., 2019; Rakovic et al., 2020a, 2020b; Raković et al., 2020), which refers to IT activities that happen outside of the IT department's radar where end users create their software.

To drive their digital transformation initiatives, organizations are increasingly considering the use of LCNC platforms as a solution to the challenge of finding qualified IT professionals. LCNC is crucial for accelerating business software development (De Silva et al., 2024), as they enable end users to create their own custom applications (Pańkowska, 2024) for automating and redesigning specific business processes (Biedova et al., 2024). Elshan et al. (Binzer et al., 2024; Elshan et al., 2024) state that the terms Low Code and No Code are often used as synonyms, but they are not the same, since No Code tools allow tools to be developed without coding.

Citizen Development (CD) is a term closely related to low code/no code development. In fact, the proliferation of citizen development has actually been enabled by low-code/no-code tools. The literature often refers to end users who develop their own applications as Citizen Developers (CD) (Biedova et al., 2024; Overeem & Jansen, 2021). CDs represent developers who have domain knowledge but limited programming skills (Mottu & Sunyé, 2024).

By enabling organizations to develop business applications with minimal training, LCDP tools democratize software development (Binzer et al., 2024; Guthardt et al., 2024). Although LCNC tools are designed for end users, certain IT skills are required to use them, i.e., IT knowledge (Rokis & Kirikova, 2023) is necessary to use these tools to their full potential. As a new, young workforce emerges that is technologically savvy and so to speak digitally native, these qualifications become less relevant (Elshan et al., 2024).

In response to the pressure of accelerated software development within budget constraints, an increasing number of organizations are embracing low code software development. According to Viljoen et al. (2023), a recent trend in software development is to become more abstract, simple, and accessible. Yet, there is a lack of empirical research and a deeper understanding of drivers and inhibitors affecting LCNC development (Käss et al., 2023b).

It is evident that this concept is becoming more popular due to the growing number of LCNC tools available on the market, and by the large IT players offering LCNC solutions, like Google, Microsoft, Amazon, Medix, Outsystems, etc. (Di Ruscio et al., 2022; Di Ruscio et al., 2022; Luo et al., 2021). Although it is difficult to compare LCNC platforms from different manufacturers, according to Phalake et al. (2024) the most popular Low Code Development Platforms (LCDP) available in the market are Mendix, Outsystem, Appians, Salesforce, Quickbase, Microsoft Power Apps and PEGA. When it comes to the Gartner Magic Quadrant for Enterprise LC Application Platforms, there are five organizations – OutSystems, Mendix, Microsoft Power Apps, Salesforce and ServiceNow – with strong capabilities in LCNC development (De Silva et al., 2024). Based on a survey of IT professionals, De Silva et al. (2024) found that Microsoft Power apps was the most commonly used LCNC platform, thanks to its impressive features such as speed, simplicity, mobile compatibility, and cost-effectiveness. In fact, Microsoft was among the first to take LCDP trends seriously and released the Power Apps platform in November 2016 (Di Ruscio et al., 2022). In January 2020, Google acquired LCDP APP Sheet and made its flagship low-code solution, while in June 2020, Amazon launched Honeycode LCDP for web and mobile application development (Di Ruscio et al., 2022). Interestingly, LC development has grown dramatically in recent years, so that it was used by less than 25% of organizations in 2020, increased to 36% the next year, and is expected to reach 70% in 2025 (Kass et al., 2022). One example of this software is Robotic Process Automation (RPA) software (Horvat et al., 2024).

Accordingly, and considering the relative novelty of the LCNC concept, three research questions were posed: RQ1: How is LCNC development defined; RQ2: What are the drivers of LCNC development, and RQ3: What are the inhibitors/challenges/limitations of LCNC development.

1. Methodology

A systematic literature review was performed to answer the previously posed research questions. This systematic literature review was conducted based on the methodology set forth by Xiao and Watson (2019) and Kitchenhaim and colleagues (Kitchenham, 2004;

Kitchenham et al., 2013). According to the aforementioned authors (Xiao & Watson, 2019), a successful review involves three major stages: planning the review, conducting the review, and reporting the review.

The first step, Planning the Review, establishes the objectives of the research, the citation bases that will be searched, the search keywords, as well as the criteria for inclusion and exclusion of papers. The research goals set based on the research questions are as follows:

- RG1: Identify the definitions of LCNC development;
- RG2: Identify the most important drivers of LCNC development; and
- RG3: Identify the most significant inhibitors/challenges/limitations of LCNC development.

For citation databases, Scopus and Web of Knowledge were chosen as the two largest databases. When searching the mentioned databases, the keywords used were "Low code" or "No code" development. The inclusion criterion must provide information to some extent on at least one of the research questions if the paper is to be included in the analysis. Among the exclusion criteria are the following: the paper is shorter than four pages, it is merely a research announcement, and it does not contribute to any of the set research goals.

Conducting the Review is the second phase of a systematic literature review. Initially, the Web of Knowledge and Scopus databases were searched using "Low code" or "No code" development keywords, followed by analysis of titles, abstracts, and keywords. A further analysis was conducted on those papers that, based on the analyzed parts, could contribute to the answer to at least one research question. Whenever the researchers were uncertain whether a paper was appropriate based on its title, abstract, or keywords, they selected it for analysis. In Table 1, the number of hits found during the primary search is shown along with the number of papers that were included in the next phase based on the initial analysis.

Table 1: Quantity of results after databases were searched by keywords

Citation database	Number of hits based on the keywords "Low code" or "No code" development	Number of papers included in second phase
Web of Knowledge	62	17
Scopus	295	43

Source: Authors' research in December 2024

In the next phase, duplicates were removed, and a unique list was created. A total of 35 papers remained after duplicates were removed, which were then downloaded in their entirety and analyzed further. Each paper was analyzed to determine if it could contribute to at least one research question. Among 35 papers, 24 can contribute to answering at least one of the three research questions. In Table 2, a list of papers with marks of usefulness is given according to the research questions. According to Figure 1, the selected papers are shown by the year of publication, indicating that the topic is very current, since the papers have been published in the last five years.

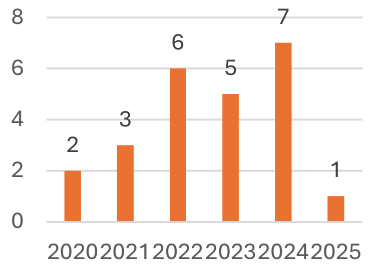
Table 2: Papers included in further analysis

Author(s)	Year	Title of the manuscript	A paper useful for a research purpose		
			Definition	Drivers	Inhibitors
Ajimati et al.	2025	Adoption of low-code and no-code development: A systematic literature review and future research agenda	✓	✓	✓
Binzer et al.	2024	Establishing a Low-Code/No-Code-Enabled Citizen Development Strategy	✓	✓	✓
De Silva et al.	2024	Role of Quality Assurance in Low-Code/No-Code Projects	✓	✓	✓
Elshan et al.	2024	Unveiling Challenges and Opportunities in Low Code Development Platforms: A StackOverflow Analysis		✓	✓
Martinez et al.	2024	Developing a novel application to digitalize and optimize construction operations using low-code technology		✓	✓
Mottu & Sunyé	2024	Emerging New Roles for Low-Code Software Development Platforms	✓	✓	✓
Pańkowska	2024	Low Code Development Cycle Investigation		✓	✓
Phalake et al.	2024	Optimization for achieving sustainability in low code development platform		✓	
Käss et al.	2023a	A Multiple Mini Case Study on the Adoption of Low Code Development Platforms in Work Systems		✓	✓
Käss et al.	2023b	Practitioners' Perceptions on the Adoption of Low Code Development Platforms		✓	✓
Martins et al.	2023	Combining low-code development with ChatGPT to novel no-code approaches: A focus-group study	✓		✓
Pinho et al.	2023	What about the usability in low-code platforms? A systematic literature review	✓		✓
Rokis & Kirikova	2023	Challenges of Low-Code/No-Code Software Development: A Literature Review	✓	✓	✓
Di Ruscio et al.	2022	Low-code development and model-driven engineering: Two sides of the same coin?		✓	✓
Gomes & Brito	2022	Low-Code Development Platforms: A Descriptive Study	✓	✓	✓
Kass et al.	2022	Drivers and Inhibitors of Low Code Development Platform Adoption		✓	✓
Rafiq et al.	2022	Understanding Low-Code or No-Code Adoption in Software Startups: Preliminary Results from a Comparative Case Study		✓	
Rokis & Kirikova	2022	Challenges of Low-Code/No-Code Software Development: A Literature Review	✓	✓	✓
Hintsch et al.	2021	Low-code development platform usage: Towards bringing citizen development and enterprise IT into harmony		✓	✓

Luo et al.	2021	Characteristics and Challenges of Low-Code Development	✓	✓	✓
Overeem & Jansen	2021	Proposing a Framework for Impact Analysis for Low-Code Development Platforms	✓	✓	
Beranic et al.	2020	Adoption and Usability of Low-Code/ No-Code Development Tools	✓	✓	✓
Sahay et al.	2020	Supporting the understanding and comparison of low-code development platforms	✓	✓	✓

Source: Authors' research

Figure1: Number of papers by years



Source: Authors' research

2. Results

2.1. Defining LCNC development

Even though the term "Low Code/No Code" development has become a buzzword in recent years, there is no a definition that is accurate or uniquely accepted. There is no doubt that Low Code/No code definition will include software development by citizen developers with minimal need for coding, while No code will enable the same but without any coding necessary. It might be helpful to see how SAP (n.d.), one of the most successful companies in the production of business software, views this or rather these two concepts:

"Low-code is a method of designing and developing applications using intuitive graphical tools and embedded functionalities that reduce traditional – or pro-code –writing requirements. Pro-code writing is still part of the development process, but low-code development offers an augmented and simplified experience to help users start creating quickly."

"No-code is a method that benefits from a similar user experience as low-code, but goes the extra mile by allowing non-technical business users to develop applications without having to write even a single line of code."

Table 3 shows the definitions of Low code and/or No code platforms that were identified in the selected papers.

Table 3: Definitions of LCNC concept

Definitions	Source
"Low-code platforms are a type of software development tool which is used for software design and development. It is based on using a graphical user interface (GUI) and pre-built elements and components, such as user interfaces, business objects, data objects (e.g., tables), and other components."	Martins et al., 2023
"LCNC development platforms are built on graphical user interface models that allow users to 'drag and drop' building blocks or visual diagrams in developing and deploying business applications."	Ajimati et al., 2025
"Low-code/no-code platforms are powerful tools that enable "citizen developers"— employees with little or no IT background—to quickly create digital solutions."	Binzer et al., 2024
"The low-code/no-code development approach is an important concept addressing current challenges in the software development."	Beranic et al., 2020
"LCD to be an approach for software development that uses tools that minimise (or eliminate) the number of lines of code written. In addition, we consider they can be enterprise solutions, conversational assistants, and visual programming tools."	Pinho et al., 2023
"A low-code development platform (LCDP) is designed for domain experts without IT skills, who want to build applications"	Mottu & Sunyé, 2024
"Low-code here means creating software with radically small amounts of code, or even without hand-coding."	Luo et al., 2021
"Low-code software development is a development approach that enhances rapid, flexible, and iterative software development by enabling quick business requirements translation through visual programming with a graphical interface, visual abstraction, and minimal hand-coding; and involving practitioners with various backgrounds and software development experience."	Rokis & Kirikova, 2023
"The surgency of terms like "low-code" or "no-code" are usually used when referring to platforms, applications, or products with a high-level programming abstraction, that are intended for end-user development (sometimes also called Citizen development) through Model-Driven Engineering (MDE) principles and that aim to serve as a tool for resolving prevailing challenges or for meeting new requirements."	Gomes & Brito, 2022
"LCDPs enable citizen developers to develop increasingly complex software without formal software development training."	Overeem & Jansen, 2021
"LC development employs a user-friendly visual interface, featuring drag-and-drop functionality and pre-built modules, offering developers a powerful tool for creating applications with a high degree of customization and flexibility."	De Silva et al., 2024
" LCDPs allow end-users with no particular programming background (called citizen developers in the LCDP jargon) to contribute to software development processes, without sacrificing the productivity of professional developers."	Sahay et al., 2020

Source: Authors' research

Based on the existing definitions, it can be concluded that the term Low Code/No code means platforms, applications, development and software products with a high level of abstraction. Platforms are tools tailored to end users, domain experts, often called citizen developers, with no or limited IT software development background. No Code represents platforms, applications, and development without any hand coding, while Low Code works with less hand coding.

2.2. LCNC drivers

Table 4 shows the drivers of LCNC development. Luo et al. (2021) note that some developers find LCD platforms easier to learn and use than programming languages, while others say that the steep learning curve makes LCD platforms a bit difficult. Furthermore, some developers consider LC development to be more affordable while others consider it to be expensive. Additionally, some developers consider LC a user-friendly tool, while others believe it requires programming expertise. While advocates for LC development assert that it creates high-quality, secure, scalable, perfectly adaptable and sustainable applications, opposition contends that LC development creates hard-to-adjust, maintain, adapt, and correct applications. Hence, the perspective of the observer and probably the tools they use are crucial factors. Therefore, certain drivers can be inhibitors at the same time (Luo et al., 2021).

The drivers of LCNC development are numerous. It is often emphasized by LCNC manufacturers how easy and quick their tools make it to develop applications. According to Ajimati et al. (2025), technology such as LCNC platforms enables companies to automate their business processes and foster a culture of innovation and collaboration between their workforces. According to the same authors, adopting LCNC and CD practices raises awareness of digital literacy in the workplace, which in turn encourages employees to acquire new skills. This will allow employees to be more autonomous and less dependent on others (Ajimati et al., 2025).

As widely known, software development involves a number of phases, including analysis, design, coding, and testing. The development of each one requires an amount of time, which can be reduced by using LC development (Phalake et al., 2024). Pańkowska (2024) also argues that LC development can serve as a prototype for further professional development.

Through LCNC development, citizen developers are empowered to become creative, engaged, and flexible (Pańkowska, 2024) as well as to acquire new skills independently, which in turn influences the attraction of talent (Biedova et al., 2024). LCNC development can also greatly reduce the use of spreadsheet applications (Biedova et al., 2024) and other forms of Shadow IT, while simultaneously bridging the gap between business and IT professionals (Ajimati et al., 2025).

Table 4: Drivers of LCNC development

Drivers	Source
Acceleration of the development cycle (time based efficiency)	Ajimati et al., 2025; Beranic et al., 2020; Biedova et al., 2024; De Silva et al., 2024; Elshan et al., 2024; Gomes & Brito, 2022; Hintsch et al., 2021; Luo et al., 2021; Martinez et al., 2024; Mottu & Sunyé, 2024; Overeem & Jansen, 2021; Phalake et

	al., 2024; Rafiq et al., 2022; Rokis & Kirikova, 2022, 2023
Expected efficiency improvements, decreased costs	Ajimati et al., 2025; Biedova et al., 2024; Kass et al., 2022; Käss et al., 2023a; Luo et al., 2021; Rokis & Kirikova, 2022, 2023
Easy to develop application (Reduction of required knowledge for application development, Newbie friendly)	Ajimati et al., 2025; Biedova et al., 2024; Di Ruscio et al., 2022; Gomes & Brito, 2022; Kass et al., 2022; Käss et al., 2023b; Luo et al., 2021; Sahay et al., 2020
Higher quality of software products	Gomes & Brito, 2022; Luo et al., 2021; Rokis & Kirikova, 2022
Reduced dependency of IT development and application development delays	Biedova et al., 2024; Binzer et al., 2024; Kass et al., 2022; Käss et al., 2023b
Replacement/reduction/mitigation of shadow IT development	Käss et al., 2023a, 2023b; Rokis & Kirikova, 2023
Easy to learn, intuitive and simple user interface	Beranic et al., 2020; Luo et al., 2021
Increased responsiveness to business and market demands	Rokis & Kirikova, 2023
Easier maintenance/customization	Biedova et al., 2024; Gomes & Brito, 2022; Rokis & Kirikova, 2023
Encouraging closer collaboration between IT and business teams	Rokis & Kirikova, 2023
Promoting digital innovations, idea development	Ajimati et al., 2025; Rokis & Kirikova, 2023
Better user experience	Luo et al., 2021
Facilitated collection of user requests	Biedova et al., 2024
Development of employees' digital skills	Ajimati et al., 2025; Biedova et al., 2024

Source: Authors

2.3. LCNC limitations, challenges and inhibitors

LCNC development faces numerous challenges or inhibitors (Rokis & Kirikova, 2022). Ajimati et al. (2025) found lack of cultural solidarity, limited organizational-wide awareness and low understanding of benefits/challenges of LCNC development. According to the same authors, the development of LCNCs is often done without awareness of cyberattacks and network security, as well as without considering data protection, quality control, and maintenance (Ajimati et al., 2025). It is common for applications to be developed without considering possible negative impacts due to rapid implementation and a lack of experience with security practices (Hintsch et al., 2021). This happens primarily because there are no rules, standards, guidelines or directives for citizen developers(Ajimati et al., 2025).

Although LCNCs are easy to use, there is still a learning curve involved (Biedova et al., 2024). Even though software companies claim that virtually any user, without a background in software development, can build an application, the reality is that citizen development still requires some level of technical literacy and a basic understanding of data modeling in order to deliver a functional digital solution (Martinez et al., 2024).

Fragmentation of platforms and vendor lock-in, that is dependency on LCNC tools manufacturers, are frequent challenges for LCNC (Ajimati et al., 2025). Furthermore, it is very difficult to identify key performance indicators and assess return on investment (Ajimati et al., 2025). Citizen developers often duplicate features and data from other applications or official IT systems (Hintsch et al., 2021). They also skip testing and documenting their software, which later negatively affects both the quality and the understanding of the software and its maintenance. It is almost impossible to build quality software without following a specific methodology, and according to Hintsch (Hintsch et al., 2021), developing a common methodology for the lifecycle of citizen development applications remains a challenge. Thus, the development of LCNC faces a number of limitations, challenges, and inhibitors. Based on the literature review, Table 5 shows the most significant ones (Hintsch et al., 2021).

Table 5: Limitations, challenges and inhibitors of LCNC development

Limitations, challenges and inhibitors	Source
Vendor lock-in/Third-party lock in (Dependence on platform vendors for software updates/security)	Ajimati et al., 2025; Biedova et al., 2024; Binzer et al., 2024; Di Ruscio et al., 2022; Kaess, 2022; Kass et al., 2022; Käss et al., 2023a, 2023b; Luo et al., 2021; Rokis & Kirikova, 2022, 2023; Sahay et al., 2020
Increased security, compliance and privacy risk (Data security breaches/cyberattacks, Shadow IT/non-compliance issues)/Security and data privacy concerns	Ajimati et al., 2025; Biedova et al., 2024; Hintsch et al., 2021; Kaess, 2022
Integration/Interoperability / Difficult to integrate to other systems	Elshan et al., 2024; Kass et al., 2022; Rokis & Kirikova, 2022; Sahay et al., 2020
Low scalability (Limited options for large scale computations/cloud resources)	Ajimati et al., 2025; Käss et al., 2023b; Rokis & Kirikova, 2022, 2023; Sahay et al., 2020
Selection of the platform	Rokis & Kirikova, 2022
Performance/Performance evaluation problem (Limited and inconsistent performance metrics)/Weaker performance	Ajimati et al., 2025; Beranic et al., 2020; Hintsch et al., 2021; Rokis & Kirikova, 2022, 2023
Lack of documentation	Biedova et al., 2024; Hintsch et al., 2021; Kass et al., 2022; Käss et al., 2023b; Martins et al., 2023
Limited testing and analysis support/ Citizen developers are not trained to test	De Silva et al., 2024; Hintsch et al., 2021; Rokis & Kirikova, 2022, 2023
Lack of flexibility and customization	De Silva et al., 2024; Kass et al., 2022; Käss et al., 2023b; Luo et al., 2021; Mottu & Sunyé, 2024; Rokis &

	Kirikova, 2022
Development knowledge gap (Lack of appropriate software skills)	Ajimati et al., 2025
No real ease of use/ A certain level of technical literacy and basic knowledge of data modeling is required / Need of basic programming knowledge	Gomes & Brito, 2022; Luo et al., 2021; Martinez et al., 2024
Lack of customization	Gomes & Brito, 2022
Duplication of features and data / Fear of island application landscape	Ajimati et al., 2025; Biedova et al., 2024; Hintsch et al., 2021; Käss et al., 2023b
Limited functionality of LCDPs / Limited freedom and creativity	Kass et al., 2022; Martins et al., 2023
Difficult estimation of total costs	Käss et al., 2023b
Lack of LCNC developers	Biedova et al., 2024; Käss et al., 2023b
Problem of IT Governance (Lack of rules/standards/guideline/directives - Knowledge integration/maintenance problems)	Ajimati et al., 2025; Kass et al., 2022; Käss et al., 2023b; Pańkowska, 2024; Sahay et al., 2020
Who is responsible for problems with LCD applications	Hintsch et al., 2021
Difficulty of maintenance and debugging	Elshan et al., 2024; Luo et al., 2021
Citizen developers engage in development, resulting in a loss of productivity	Biedova et al., 2024
Role conflicts and collaboration (Loss of control/reduced influence of IT professionals on business professionals)	Ajimati et al., 2025; Biedova et al., 2024
Lack of application development planning / Lack of strategic thinking about architecture	Ajimati et al., 2025; Biedova et al., 2024; Rokis & Kirikova, 2023
No access to source code	Rokis & Kirikova, 2022, 2023
Requirement Specification	Rokis & Kirikova, 2022
High prices	Luo et al., 2021
Insufficient transparency, platform capabilities are not understood by users	Pinho et al., 2023

Source: Authors' research

Conclusion and limitations

Digital workplace (Raković et al., 2022) cannot be imagined without tools that facilitate process transformation within organizations. As a result of their agility and ease of use, LCNC platforms (Martins et al., 2023) have become increasingly popular in both business and academia (Hagel et al., 2024). LCNC development is particularly interesting in domains that need to automate business processes, but practitioners have very different and conflicting views on the advantages and disadvantages of this type of software development (Luo et al., 2021).

Although LCNC has been studied, Kass et al. (2022) call for further empirical research in order to fill the empirical gap. Furthermore, LCNC tool manufacturers promise significant improvements, but there is little evidence in the literature (Varajão et al., 2023). Varajão et al. (2023) wonder if this is just propaganda from LCNC development tool providers, because their advertisements seem too good to be true, and believe that it is important to distinguish between what is advertising and what is feasible.

According to Binzer et al. (2024), introducing and using LCNC tools requires a holistic strategy, since installing these platforms will not ensure success. Without a strategic approach, they can result in inefficiency, unachieved goals, and missed investments. There are many organizations without a strategy, and it is important to balance the autonomy of Central Development with centralized governance, which ensures that tools are adopted, integrated, and managed throughout the organization (Binzer et al., 2024). In some cases, low code/no code tool creators can mitigate certain challenges by improving documentation and instructions (Rokis & Kirikova, 2022).

Organizations should be very careful when choosing and adopting LCNC platforms. Most often, however, citizen developers make the decision to adopt LCNC on their own without consulting IT department (Biedova et al., 2024). As a result, citizen developers often make these decisions in a hurry, frustrated by the urgent need for certain applications to operate (Biedova et al., 2024).

Ajimati et al. (2025) advocate observing and managing LCNC development holistically, and evaluating contracts and platforms carefully to avoid vendor dependency and costs of switching to other platforms. The organization's current systems should also be considered when choosing platforms. Therefore, Microsoft Power is a good choice for organizations that have already implemented other Microsoft applications, but not for those that use other ecosystems (Binzer et al., 2024). In addition to the LCNC platform itself, it is important to consider whether citizen developers are developing scalable and durable LCNC applications (Viljoen et al., 2023). Naturally, citizen developers need to be supported by carefully chosen LCNC platforms and trained on how to use them (Biedova et al., 2024).

It is clear that the use of artificial intelligence will facilitate the easier and faster development of LCNC applications by citizen developers (Hagel et al., 2024; Martins et al., 2023). Furthermore, McHugh et al. (2024) emphasize the need to establish citizen development/LCNC programs in secondary schools in order to train as many users as possible for this method of software development.

LCNC application development is often found at the intersection of Shadow IT and End User Development. Shadow IT refers to the creation of applications without the knowledge of the IT department, so LCNC application development that is not approved by the IT department can also be considered as Shadow IT. On the other hand, End User Development refers to application development by users who do not have adequate IT knowledge, so if users who engage in LCNC application development lack the necessary IT skills, they can be considered End User Developers.

Despite addressing the topics of definition, drivers, and inhibitors of LCNC development in detail, the paper has some limitations. A limitation of the study was that only two (albeit the largest) citation databases were investigated, along with the fact that the reviewed papers were published in English. Furthermore, the research results have not been practically confirmed. As a result, the following directions can be considered for future research: an analysis of papers published in journals and symposia indexed in other citation databases, and conducting case studies and surveying business and IT users.

References

Ajimati, M. O., Carroll, N., & Maher, M. (2025). Adoption of low-code and no-code development: A systematic literature review and future research agenda. *Journal of Systems and Software*, *222*, 112300. Doi: https://doi.org/10.1016/j.jss.2024.112300

Beranic, T. , Rek, P., & Hericko, M. (2020). Adoption and Usability of Low-Code/ No-Code Development Tools. *Proceedings of the Central European Conference on Information and Intelligent Systems*, 97–103.

Biedova, O., Ives, B., Male, D., & Moore, M. (2024). Strategies for Managing Citizen Developers and No-Code Tools. *MIS Quarterly Executive*, *23*(2), 165–183. Doi: https://doi.org/10.17705/2msqe.00093

Binzer, B., Elshan, E., Fürstenau, D., & Winkle, T. J. (2024). Establishing a Low-Code/No-Code-Enabled Citizen Development Strategy. *MIS Quarterly Executive*, *23*(3), 253–273. Doi: https://doi.org/10.17705/2msqe.00097

De Silva, D. I., Shangavie, R., & Ranathunga, R. A. A. L. (2024). Role of Quality Assurance in Low-Code/No-Code Projects. *2024 International Conference on Information Networking (ICOIN)*, 789–794. Doi: https://doi.org/10.1109/ICOIN59985.2024.10572203

Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., & Wimmer, M. (2022). Correction to: Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, *21*(5), 1687–1687. Doi: https://doi.org/10.1007/s10270-022-01038-5

Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., & Wimmer, M. (2022). Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, *21*(2), 437–446. Doi: https://doi.org/10.1007/s10270-021-00970-2

Đorđević Milutinović, L., Raković, L., & Antić, S. (2023). Characteristics of Spreadsheet-Based Shadow IT in Serbian Companies. In M. Mihić, S. Jednak, & G. Savić (Eds.), *Sustainable Business Management and Digital Transformation: Challenges and Opportunities in the Post-COVID Era. SymOrg 2022. Lecture Notes in Networks and Systems, vol 562.* (pp. 148–171). Springer. Doi: https://doi.org/10.1007/978-3-031-18645-5_10

Elshan, E., Siemon, D., Bruhin, O., Kedziora, D., & Schmidt, N. (2024). Unveiling Challenges and Opportunities in Low Code Development Platforms: A StackOverflow Analysis. *Hawaii International Conference on System Sciences 2024*, 7269–7279.

Gomes, P. M., & Brito, M. A. (2022). Low-Code Development Platforms: A Descriptive Study. *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–4. Doi: https://doi.org/10.23919/CISTI54924.2022.9820354

Guthardt, T., Kosiol, J., & Hohlfeld, O. (2024). Low-code vs. the developer: An empirical study on the developer experience and efficiency of a no-code platform. *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, 856–865. Doi: https://doi.org/10.1145/3652620.3688332

Hagel, N., Hili, N., & Schwab, D. (2024). Turning Low-Code Development Platforms into True No-Code with LLMs. *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, 876–885. Doi: https://doi.org/10.1145/3652620.3688334

Hintsch, J., Staegemann, D., Volk, M., & Turowski, K. (2021). Low-code development platform usage: Towards bringing citizen development and enterprise IT into harmony. *Australasian Conference on Information Systems, ACIS 2021*.

Horvat, D., Ivanišević, R., & Gluščević, L. (2024). Risks associated with robotic process automation. *Journal of Process Management and New Technologies*, *12*(1–2), 72–82. Doi: https://doi.org/10.5937/jpmnt12-50617

Kaess, S. (2022). Low Code Development Platform Adoption: A Research Model. *ACIS 2022 Proceedings, Australasian Conference on Information Systems*.

Kass, S., Strahringer, S., & Westner, M. (2022). Drivers and Inhibitors of Low Code Development Platform Adoption. *2022 IEEE 24th Conference on Business Informatics (CBI)*, 196–205. Doi: https://doi.org/10.1109/CBI54897.2022.00028

Käss, S., Strahringer, S., & Westner, M. (2023a). A Multiple Mini Case Study on the Adoption of Low Code Development Platforms in Work Systems. *IEEE Access*, *11*, 118762–118786. Doi: https://doi.org/10.1109/ACCESS.2023.3325092

Käss, S., Strahringer, S., & Westner, M. (2023b). Practitioners' Perceptions on the Adoption of Low Code Development Platforms. *IEEE Access*, *11*, 29009–29034. Doi: https://doi.org/10.1109/ACCESS.2023.3258539

Kitchenham, B. (2004). *Procedures for performing systematic reviews*. Keele University. https://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2013). Systematic literature reviews in software engineering – a systematic literature review. *Information and Software Technology*, *51*(1), 7–15.

Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). Characteristics and Challenges of Low-Code Development. *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–11. Doi: https://doi.org/10.1145/3475716.3475782

Martinez, E., Pfister, L., & Stauch, F. (2024). Developing a novel application to digitalize and optimize construction operations using low-code technology. *41st International Symposium on Automation and Robotics in Construction (ISARC 2024)*, 283–290. Doi: https://doi.org/10.22260/ISARC2024/0038

Martins, J., Branco, F., & Mamede, H. (2023). Combining low-code development with ChatGPT to novel no-code approaches: A focus-group study. *Intelligent Systems with Applications*, *20*, 200289. Doi: https://doi.org/10.1016/j.iswa.2023.200289

McHugh, S., Carroll, N., & Connolly, C. (2024). Low-Code and No-Code in Secondary Education—Empowering Teachers to Embed Citizen Development in Schools. *Computers in the Schools*, *41*(4), 399–424. Doi: https://doi.org/10.1080/07380569.2023.2256729

Modupe, A. (2023). Digital transformation and firm efficiency in the Nigerian manufacturing sector. *Ekonomski Horizonti*, *25*(3), 215–230. Doi: https://doi.org/10.5937/ekonhor2302215A

Mottu, J.-M., & Sunyé, G. (2024). Emerging New Roles for Low-Code Software Development Platforms. *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, 905–914. Doi: https://doi.org/10.1145/3652620.3688337

Overeem, M., & Jansen, S. (2021). Proposing a Framework for Impact Analysis for Low-Code Development Platforms. *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 88–97. Doi: https://doi.org/10.1109/MODELS-C53483.2021.00020

Pańkowska, M. (2024). Low Code Development Cycle Investigation. In X.-S. Yang, S. Sherratt, N. Dey, & A. Joshi (Eds.), *Lecture Notes in Networks and Systems 1011* (pp. 265–275). Springer. Doi: https://doi.org/10.1007/978-981-97-4581-4_19

Phalake, V. S., Joshi, S. D., Rade, K. A., Phalke, V. S., & Molawade, M. (2024). Optimization for achieving sustainability in low code development platform. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, *18*(8), 5717–5724. Doi: https://doi.org/10.1007/s12008-023-01338-0

Pinho, D., Aguiar, A., & Amaral, V. (2023). What about the usability in low-code platforms? A systematic literature review. *Journal of Computer Languages*, *74*, 101185. Doi: https://doi.org/10.1016/j.cola.2022.101185

Popović, A. (2020). Implications of the Fourth Industrial Revolution on sustainable development. *Economics of Sustainable Development*, *4*(2), 45–60. Doi: https://doi.org/10.5937/ESD2001045P

Prinz, N., Huber, M., Leonhardt, J., & Riedinger, C. (2024). Unleash the Power of Citizen Development: Leveraging Organizational Capabilities for Successful Low-Code Development Platform Adoption. *Hawaii International Conference on System Sciences 2024*, 569–578.

Rafiq, U., Filippo, C., & Wang, X. (2022). Understanding Low-Code or No-Code Adoption in Software Startups: Preliminary Results from a Comparative Case Study. In D. Taibi, M. Kuhrmann, T. Mikkonen, J. Klünder, & P. Abrahamsson (Eds.), *Product-Focused Software Process Improvement. PROFES 2022. Lecture Notes in Computer Science, vol 13709* (pp. 390–398). Springer. Doi: https://doi.org/10.1007/978-3-031-21388-5_27

Raković, L. (2020). Shadow IT – Systematic Literature Review. *Information Technology And Control*, *49*(1), 144–160. Doi: https://doi.org/10.5755/j01.itc.49.1.23801

Rakovic, L., Duc, T. A., & Vukovic, V. (2020a). Shadow IT and ERP: Multiple Case Study in German and Serbian Companies. *Journal of East European Management Studies*, *25*(4), 730–752. Doi: https://doi.org/10.5771/0949-6181-2020-4-730

Rakovic, L., Duc, T. A., & Vukovic, V. (2020b). Shadow IT and ERP: Multiple Case Study in German and Serbian Companies. *Journal of East European Management Studies*, *25*(4), 730–752. Doi: https://doi.org/10.5771/0949-6181-2020-4-730

Raković, L., Sakal, M., & Matković, P. (2019). Cluster analysis of the frequency of use of spreadsheet functionalities. *Anali Ekonomskog Fakulteta u Subotici*, *41*, 95–111. Doi: https://doi.org/10.5937/AnEkSub1941095R

Rakovic, L., Sakal, M., & Matkovic, P. (2022). Digital workplace–advantages and challenges. *Anali Ekonomskog fakulteta u Subotici*, *58*(47), 65-78. Doi: https://doi.org/10.5937/AnEkSub2247065R

Raković, L., Sakal, M., Matković, P., & Marić, M. (2020). Shadow IT – Systematic Literature Review. *Information Technology And Control*, *49*(1), 144–160. Doi: https://doi.org/10.5755/j01.itc.49.1.23801

Rokis, K., & Kirikova, M. (2022). Challenges of Low-Code/No-Code Software Development: A Literature Review. In Ē. Nazaruka, K. Sandkuhl, & U. Seigerroth (Eds.), *Perspectives in Business Informatics Research. BIR 2022. Lecture Notes in Business Information Processing* (Vol. 462, pp. 3–17). Springer. Doi: https://doi.org/10.1007/978-3-031-16947-2_1

Rokis, K., & Kirikova, M. (2023). Exploring Low-Code Development: A Comprehensive Literature Review . *Complex Systems Informatics and Modeling Quarterly*, *36*, 6886. Doi: https://doi.org/10.7250/csimq.2023-36.04

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *2020 46th*

Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 171–178. Doi: https://doi.org/10.1109/SEAA51224.2020.00036

SAP. (n.d.). *What is low-code/no-code application development?* Retrieved February 28, 2025, from https://www.sap.com/products/technology-platform/build/what-is-low-code-no-code.html

Schenkenfelder, B., Brandstätter, U., Kirchtag, H., & Wimmer, M. (2024). Low-Code Development and End-User Development: (How) Are They Different? *Proceedings of the First International Workshop on Participatory Design & End-User Development - Building Bridges (PDEUD2024)*.

Ubiparipović, B., Raković, L., Marić, S., & Vuković, V. (2023). Digital business agility. *Ekonomika*, *69*(2), 75–86. Doi: https://doi.org/10.5937/ekonomika2302075U

Varajão, J., Trigo, A., & Almeida, M. (2023). Low-code Development Productivity - "Is winter coming" for code-based technologies? *Admqueue*, *21*(5).

Viljoen, A., Nguyen, J., Kauschinger, M., & Hein, A. (2023). Fostering Scalable Citizen Development in Organizations: Towards a Guiding Framework. *Twenty-Ninth Americas Conference on Information Systems - AMCIS 2023*.

Xiao, Y., & Watson, M. (2019). Guidance on Conducting a Systematic Literature Review. *Journal of Planning Education and Research*, *39*(1), 93–112. Doi: https://doi.org/10.1177/0739456X17723971