Management Review Quarterly
https://doi.org/10.1007/s11301-026-00603-2

The democratization of software engineering: evolution, definition, and the future of low-code and no-code platforms

Adrian Abendroth[1]

Received: 27 November 2025 / Accepted: 13 April 2026
© The Author(s) 2026

Abstract
As organizations face increasing pressure for digital agility, Low-Code and No-Code Platforms (LCP/NCP) have emerged as a primary drivers for the democratization of software development. However, the current lack of terminological consensus complicates the synthesis of research findings and obscures strategic decision-making in practice. This paper addresses this conceptual fragmentation by conducting a Systematic Literature Review (SLR) and graph-based analysis of 383 definitions from 306 publications published between 2014 and 2025. Through word frequency and conceptual synthesis, the study establishes a consistent conceptual foundation for the field. It characterizes LCPs as process-oriented environments that enable both professional and citizen developers to leverage model-based approaches and minimal coding. Conversely, NCPs are defined as platforms that empower users without programming knowledge by eliminating code-based programming entirely through visual, drag-and-drop interfaces. The analysis further identifies a shift from instruction-based to intent-based development, influenced by emerging trends such as agentic-AI and "vibe coding". By mapping 10 identified system classes across a functional taxonomy, this research offers a strategic framework for researchers to build a cumulative body of knowledge and for practitioners to mitigate the risks of shadow IT and technical debt in the era of hyperautomation.

Keywords Low-code platforms (LCP) · No-code platforms (NCP) · Systematic literature review (SLR) · Definitions · Software democratization

Communicated by Joern Block.

✉ Adrian Abendroth
adrian.abendroth@wi.uni-potsdam.de

[1] Chair of Business Informatics, esp. Processes and Systems, University of Potsdam, 14482 Potsdam, Germany

Published online: 08 June 2026

1 Introduction

Low-Code and No-Code platforms (LCP/NCP) are integrated toolsets designed to empower both professional developers and non-developers to rapidly create business applications with minimal effort (Adrian et al. 2020; Bock & Frank 2021; Rymer et al. 2019; Waszkowski 2019). These platforms significantly reduce reliance on traditional coding by enabling users to develop applications through visual and declarative tools, such as drag-and-drop interfaces and graphical templates (Bock & Frank 2021; Metrôlho et al. 2019; Rymer et al. 2019; Sahay, Indamutsa et al., 2020; Totterdale 2018).

As these platforms democratize software development, their exploration represents a relatively new but rapidly expanding field in both industry and academia, as demonstrated by a substantial increase in related publications over recent years (Goasduff 2021). Reflecting this growth, analyst firms such as Gartner forecast that by year 2026 over 80% of organizations will be utilizing LCP/NCP in some capacity (Gartner 2022). The global LCP market is forecast to amount to approximately 65 billion U.S. dollars by 2027 (Vailshery 2024). Beyond accelerating application creation, LCP/NCP are essential drivers of hyperautomation, the technology-driven, end-to-end automation of complex business processes within digital transformation efforts (Piridi et al. 2025). Hyperautomation integrates technologies such as Robotic Process Automation (RPA), Machine Learning (ML), and Artificial Intelligence (AI), with LCP/NCP serving as a key enabler (Haleem et al. 2021; Piridi et al. 2025). Recent advancements such as agentic-AI (Matvitskyy et al. 2025), vibe coding, and other emerging trends further extend the potential of LCP/NCP in automation strategies when working together (Gadde 2025; Ray 2025). The consideration of new technologies for LCP/NCP is essential, as they continuously redefine automation capabilities and ensure that organizations remain competitive. Research has already examined various aspects of LCP/NCP, including their technical features, comparative analyses, drivers, and inhibitors (Käss et al. 2022) as well as their feasibility and usability (Pinho et al. 2023). Moreover, research has encompassed a multitude of applications, including ML (Li & Wu 2022) the construction industry (Martinez & Pfister 2023), small and medium-sized enterprises (SMEs), and blockchain technology (Curty et al. 2023). Despite these endeavors, the integration of this knowledge across multiple domains remains fragmented.

However, the overarching concept of LCP/NCP remains abstract and challenging to grasp, with multiple interpretations complicating the terminology (Pinho et al. 2023). This conceptual ambiguity is not merely a linguistic issue but a fundamental barrier to the field's maturity. This research derives its importance from addressing the two-fold challenge created by this conceptual fragmentation. First, for the academic community, the absence of a coherent understanding and a standardized conceptual framework significantly hampers the advancement of research in this area. Without uniform definitions, the researchers faces difficulties in replicating studies, comparing findings, and building a cumulative body of knowledge, rendering much of the existing literature siloed and non-transferable. Second, for organizational decision-makers, this terminological confusion creates significant strategic risks in procurement and governance. Without a clear distinction between platform capabili-

ties, organizations risk investing in technologies that are ill-suited for their specific functional requirements or technical debt profiles.

It is therefore critical to establish a common understanding of LCP/NCP to enhance communication and collaboration among researchers and to foster the development of robust theoretical models and practical applications. This paper aims to address these issues by offering a comprehensive understanding of what precisely defines LCP/NCP, how definitions change over time and how new technology advances relevant for both definitions. The divergences in understanding and the disparate nature of existing research emphasize the existing gaps in the field. To address these gaps, the following research questions (RQ) were formulated:

- RQ1: Which scientific and science related definitions of the terms LCP and NCP do exist?
- RQ2: How do these definitions change over time?
- RQ3: How can the term LCP and NCP be uniformly defined?
- RQ4: What differentiates LCP and NCP?

This paper begins by introducing and categorizing the underlying technologies and background concepts that enable LCP/NCP (chapter 2). It then outlines the research methodology, detailing the scope of the study, the database search strategy, criteria for literature selection, and the analytical approach applied to the selected works (chapter 3). The subsequent section presents the findings of the literature review, focusing on the definitions of LCP and NCP identified (chapter 4), followed by an examination of how these definitions have evolved over time (chapter 5). Based on this analysis, new definitions are proposed (chapter 6) and then systematically compared with existing literature to identify key differentiations (chapter 7). Afterwards, the results are synthesized by mapping identified platforms across a functional taxonomy, analyzing organizational and technical implications, and proposing a strategic research agenda for the field (chapter 8). Finally, the paper concludes by discussing the implications for research and practice, addressing the study's limitations, and summarizing the core contributions and future directions for the field (chapter 9).

2 Theoretical background

LCP and NCP are the result of a long evolution in software development methods, languages, and tooling, aimed at increasing productivity, reducing the need for manual coding, and making application development accessible to a wider range of users (Bucaioni et al. 2022). This historical progression is anchored in the diverse underlying technical foundations illustrated in Fig. 1. To understand their conceptual and technical foundations, it is necessary to revisit several key developments in programming language generations, software engineering approaches, and end-user development (EUD) paradigms.

An important perspective on this evolution is provided by the concept of the low-code iceberg (Bucaioni et al. 2022). This metaphor highlights that what is most visible to end users and citizen developers are the simplified interfaces and abstractions

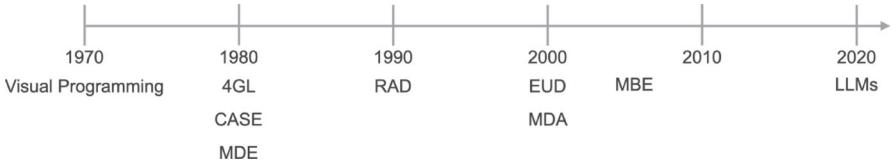

Fig. 1 Underlying technical Foundation of LCP/NCP

offered by LCPs and NCPs. However, beneath this surface lies a large body of foundational technologies and prior developments, including model-driven engineering, workflow management systems, visual programming, and integration mechanisms, which are discussed in the following sections.

2.1 Visual programming languages (VPL)

VPL trace their origins back to the 1970s, with early developments taking place at research centers such as Xerox PARC (Skog 2005). VPLs enable programming through the visual manipulation of elements like boxes, diagrams and arrows, rather than purely textual code (Jost et al. 2014). They combine visual expressions with the structured layout of text and graphics to facilitate the creation of software (Sanchis et al. 2020). In contrast to LCP, VPLs focus primarily on graphical representations such as boxes, arrows, and other diagrammatic elements (Bragg & Driskill 1994). LCP, by comparison, are designed specifically for building applications and typically integrate VPL as a front-end layer, supported by a more comprehensive development layer that manages application logic, integration, and deployment (Sanchis et al. 2020).

2.2 Programming language generations and 4GL

From the first generation (1GL) of machine languages in the late 1940s (Craven 2016; Hughes 2002) to the fourth generation of programming languages (4GL) in the 1980s, the history of programming has been marked by a continuous rise in abstraction levels (IEEE 1990; Kahanwal 2013; Nagy et al. 2011). While second-generation languages used one-to-one translation to machine instructions with assembly (IEEE 1990; Kahanwal 2013), third-generation languages such as C, Java, or Pascal enabled procedural and object-oriented programming (IEEE 1990; Kahanwal 2013), 4GLs further increased productivity by offering database management, query languages, report generators, and visual design tools (IEEE 1990; Kahanwal 2013; Nagy et al. 2011). Many 4GL systems, such as SQL-based platforms, LCPs like FileMaker or NCPs like Bubble, made application creation more accessible to non-programmers and are using concepts like declarative design and embedded visual modelling (Nagy et al. 2011).

2.3 Computer-aided software engineering (CASE) tools

In the 1980s, CASE tools emerged to support specific tasks within the software production process, with the broader goal of covering the entire software development lifecycle (Lundell & Lings 2004). These tools often took the form of integrated workbenches that combined multiple development utilities into a single environment. Many were object-oriented and supported visual modeling through notations such as UML (Fuggetta 1993; Tarasiev et al. 2018). However, their adoption remained limited because they were costly in terms of both software licensing and training (Orlikowski 1993), often failed to fully address actual user requirements (Lundell & Lings 2004) or are high complex (Sanchis et al. 2020).

Although adoption was more limited than initially anticipated, CASE is a fundament for the integrated development environments and model-based editors found in modern LCP/NCP, particularly with their focus on object-oriented modeling and automated code generation (Cabot 2020; Martinez & Cisterna 2023).

2.4 Rapid application development (RAD)

The RAD methodology, introduced in the early 1990s, is an information systems development approach aimed at delivering high-quality systems quickly and at lower cost (Beynon-Davies & Williams 2003). The term was first introduced by James Martin in 1991 (Martin 1991) and builds on Barry Boehm's Spiral Model (Borchers 2018). RAD employs an iterative development process emphasizing high-level phases that revolve around rapid prototyping to refine requirements and solutions. Rather than prescribing a fixed set of techniques, RAD allows for flexible adaptation of methods to suit the needs of individual projects (Stapleton 1997). RAD influenced LCP/NCP by embedding iterative design cycles, quick testing, and close user collaboration directly into development environments, thereby reducing time-to-market (Lebens 2025).

2.5 Model-driven engineering (MDE) and related paradigms

A core technical foundation of LCP/NCP is MDE starting in the 1980s (Coutaz 2010), which treats models as primary, machine-processable artifacts that drive the engineering process (Cabot 2020; Hailpern & Tarr 2006). Model-Driven Development (MDD), applies MDE principles to software development (Hailpern & Tarr 2006), meaning that instead of writing most of the system directly in code, developers create high-level models (such as diagrams or domain-specific descriptions) that are then automatically transformed into executable software or configuration artifacts. Model-Driven Architecture (MDA) is a specific MDD version defined by the Object Management Group (OMG) (Hailpern & Tarr 2006). Model-Based Engineering (MBE), starting in the 1990s and getting traction in late 2000s (Akundi & Mondragon 2022; Haskins 2011), uses models support rather than drive the process (Krouwel et al. 2022). In LCP/NCP, MDE principles underpin visual modelling, automatic code generation, and declarative development.

2.6 End-user development (EUD) and citizen development

EUD arose in the 2000s extends software creation capabilities to non-professional developers, enabling them to design, adapt, and maintain applications according to their domain-specific needs (Lieberman et al. 2006). It involve the whole software development lifecycle, including design and use, and goes beyond programming activities (Barricelli et al. 2019). It distinguished itself from design by involving users in the whole software lifecycle (Simonsen & Robertson 2013). A subset is End-User Programming (EUP) enabling end users to create their own programs (Ko et al. 2011) by writing programs using special-purpose or professional programming languages such as C, Java or spreadsheets. End-User Software Engineering (EUSE) add programming capabilities and systematic engineering practices to EUD enhancing debugging support, maintainability, and version control (Burnett 2009).

In the context of LCP/NCP, this evolution manifests in the concept of the "citizen developer", a business user with minimal coding skills who can create applications through visual and model-driven tools (Carroll & Ajimati 2024; Di Ruscio et al., 2022; Fattah-Weil et al. 2025; Sahay, Di Ruscio, et al., 2020). A modality within this evolution is Programming by Demonstration (PBD), where users define logic not by writing rules, but by performing a task that the system generalizes into a reusable program, lowering the cognitive barrier for citizen developers (Hirzel 2023; Liu et al. 2025).

2.7 Emerge of supplementary technologies

The evolution of LCP and NCP has not occurred in isolation but has been closely intertwined with the advancement of complementary technologies that strengthen their automation, integration, and adaptability capabilities. As organizations increasingly pursue end-to-end process automation and digital transformation, these supplementary technologies play a crucial role in expanding the functional scope of LCP/NCP and enabling their seamless operation within complex enterprise environments.

Workflow management systems (WfMS) comprise a modeling environment that enables the design of workflows, an administration client that supports monitoring and control by workflow administrators, and user clients that provide access for end users as well as integration with other applications (Gronau 2022, p. 295). LCP frequently integrate WfMS functionalities, since workflows represent a critical component. In this context, a workflow engine is typically incorporated together with a modeling environment, enabling both the orchestration and the monitoring of workflows (Bock & Frank 2021; Kokala 2022). With WfMS components in LCP, often non-technical users can bridge the gap between IT and business users empowering domain experts to contribute directly to BPM (Kokala 2022).

Robotic Process Management (RPA) represents another technology that can be integrated into LCPs. RPA is particularly useful for automating repetitive manual tasks that occur when system interfaces are missing. In such cases, software robots can, for instance, transfer data from one system to another, thereby reducing the need for human intervention in routine processes. While LCPs can be applied to the development of entirely new business applications, the integration of RPA focuses on auto-

mating existing manual activities and bridging gaps where system interoperability is not available (Noack & Fedtke, 2025, p. 43).

2.8 Artificial Intelligence (AI) and natural language

AI, especially large-language models (LLM), introduces new interaction paradigms in LCP, particularly through Programming by Natural Language (PBNL). These prompt-based approaches, increasingly characterized in industry discourse as vibe coding, enable users to generate applications by formulating natural language inputs via text or speech, which are then used to automatically create both the user interface and the underlying business logic in an initial step (Hirzel 2023; Kokala 2022; Liu et al. 2025). The automatically generated proposal can subsequently be adapted and refined by the user, allowing for an iterative and user-driven development process. This approach integrates conversational interactions as a central mechanism for building applications, further simplifying application design and development (Lebens 2025; Stapleton 1997). Beyond application generation, AI-powered workflow automation supports predictive analytics (Bender et al. 2024) by anticipating potential process deviations or optimization opportunities. Machine learning techniques further enhance rule engines by analyzing historical data, thereby increasing the adaptability and accuracy of business logic (Kokala 2022).

2.9 Synthesis of technical foundations

Taken together, these developments converge in modern LCP/NCP. They combine high-level abstractions from 4GL, automation and modelling from CASE and MDE, iterative delivery from RAD, and accessibility features from EUD and VPL. On a conceptual level, this evolution enables a dual-user ecosystem where professional developers utilize model-driven frameworks for architectural standardization, while citizen developers leverage VPL and PBD to democratize application creation.

Contemporary LCP/NCP are no longer isolated tools but integrated environments enriched by WfMS for orchestration and LLMs for natural-language synthesis. The emergence of PBNL and vibe coding represents the most recent layer of the low-code iceberg: a conversational interface that abstracts away the complexity of the underlying CASE and MDE foundations (Bucaioni et al. 2022; Hirzel 2023; Liu et al. 2025). Furthermore, AI agents extend these platforms by supporting adaptive and autonomous behaviors, thereby broadening the scope of LCP (Awasi 2025).

3 Methodology

To address the research questions at hand, relevant data was gathered, by conducting a structured literature review (SLR) using the methodology proposed by vom Brocke et al. (2009). In adhering to the principles outlined by Cooper (1988), the SLR has been structured to ensure integration, neutrality, representative coverage, and historical, conceptual organization (Fig. 2). This paper has adopted a neutral perspective, catering to both specialized and practitioners within the field of information systems

Characteristic		Categories			
(1)	Focus	Research outcome	Research methods	Theories	Application
(2)	Goal	Integration	Criticism	Identification of central issues	
(3)	Perspective	Neutral representation	Espousal of position		
(4)	Coverage	Exhaustive	Exhaustive and selective	Representative	Central/Pivotal
(5)	Organization	Historical	Conceptual	Methodological	
(6)	Audience	Specialized scholars	General scholars	Practitioners	General public

Fig. 2 Taxonomy of the SLR

and business informatics interested in LCP/NCP. The goal of this SLR is to integrate the various definitions into two single definitions of LCP and NCP in a more accurate way.

In this SLR, definitions and descriptions of the term LCP and NCP are systematically retrieved from the literature and presented to answer RQ1. To answer RQ2-4, the definitions and descriptions are then systematically analyzed for the occurrence of specific terms as shown in the definition analysis of Girard and Girard (2015) and Ritterbusch and Teichmann (2023), in order to establish new holistic definition of LCP and NCP and compared with existing literature.

3.1 Keyword and database definition

To comprehensively survey the literature, a systematic search across the following nine bibliographic databases was conducted: ACM digital library, AIS electronic library, dblp, ieeeexplore, ProQuest, ScienceDirect, Springerlink, Web of Science and Wiley online library. To query these databases a search string was developed and as required, tailored for each database:

‘low-code’ OR ‘low code’ OR ‘no code’ OR ‘no-code’.

The search string was performed at the title level only to ensure relevance to the LCP/NCP domain.

3.2 Inclusion and exclusion criteria

Studies were included if both of the following criteria were met:

(IC1) The study focuses on the field of LCP/NCP, either conceptually or through the development of a specific solution using LCP/NCP; (IC2) the study was published in a peer-reviewed journal, conference proceedings, made available through scientific repositories or a report from an company institution.

Studies were excluded based on the following criteria:

(EC1) The publication is a thesis or dissertation; (EC2) is written in a language other than English or German; (EC3) does not contain a self-made or extended defini-

tion of LCP or NCP; and (EC4) only cites existing definitions without modifying or extending them.

In this context, a self-made definition refers to a definition newly formulated by the authors, based on their own reasoning or analysis. An extended definition refers to a definition that builds on existing work by modifying, refining, or elaborating on a previously published definition.

3.3 Conduction the search process

The search was initially conducted over a four-week period in April 2024 and was subsequently updated in early February 2026 to ensure the inclusion of the most recent scholarly contributions and technological advancements. A total of 1705 relevant papers were obtained from the search process. In this context, each obtained paper is referred to as a'hit'. The preliminary selection consists of hits that met the criteria for inclusion in the study.

Initially, the database search produced a set of hits, where each hit represents a publication identified during the search process across the databases. As a result, the preliminary collection of literature consisted of 1705 hits. After removing duplicates, the remaining record titles (n = 1153) were screened, and ineligible records (n = 454) unrelated to the topic of LCP/NCP were excluded. After reviewing the abstracts of the remaining records (n = 699), 209 ineligible records were excluded due to their focus on adjoining topics like process modeling related to LCP/NCP. Furthermore, efforts were made to gather full texts (n = 490), with some unavailable (n = 72). After reviewing the full texts of the remaining reports (n = 418), 186 final hits were identified.

To finalize the selection of 186 final hits, a backward search was performed using the tool citationchaser (Haddaway et al. 2021). The backward search resulted in 4122 hits after removing duplicates. After screening titles and abstracts with inclusion and exclusion criteria (called hot pics), 315 hot pics remained. The full texts of the remaining 315 search results were carefully reviewed, resulting in the identification of 64 final hits (Fig. 3).

Employing the same methodology, a forward search in Google Scholar with initial 4651 hits was conducted. Titles and abstracts were screened, resulting in the elimination of 4559 papers. Full-text screening of the remaining 92 hot pics identified 56 additional final hits. In total, the SLR yielded 306 final articles. Out of the 306 final articles, a total of 383 definitions were extracted: 286 for LCP and 97 for NCP. While 77 articles defined both definitions, 209 articles only defined LCP and 20 article only defined NCP (Fig. 4).

3.4 Data preparation

For the analysis, a table was constructed to encompass the type of definition (own, extended), description of the definition, used source references for the extended definitions as well as various metadata fields, including publication title, year published and outlet. If mentioned, differences and similarities between LCP and NCP where noted. To support a subsequent network analysis, it was also recorded which definitions referenced others within the final pool of 306 studies. This allowed for

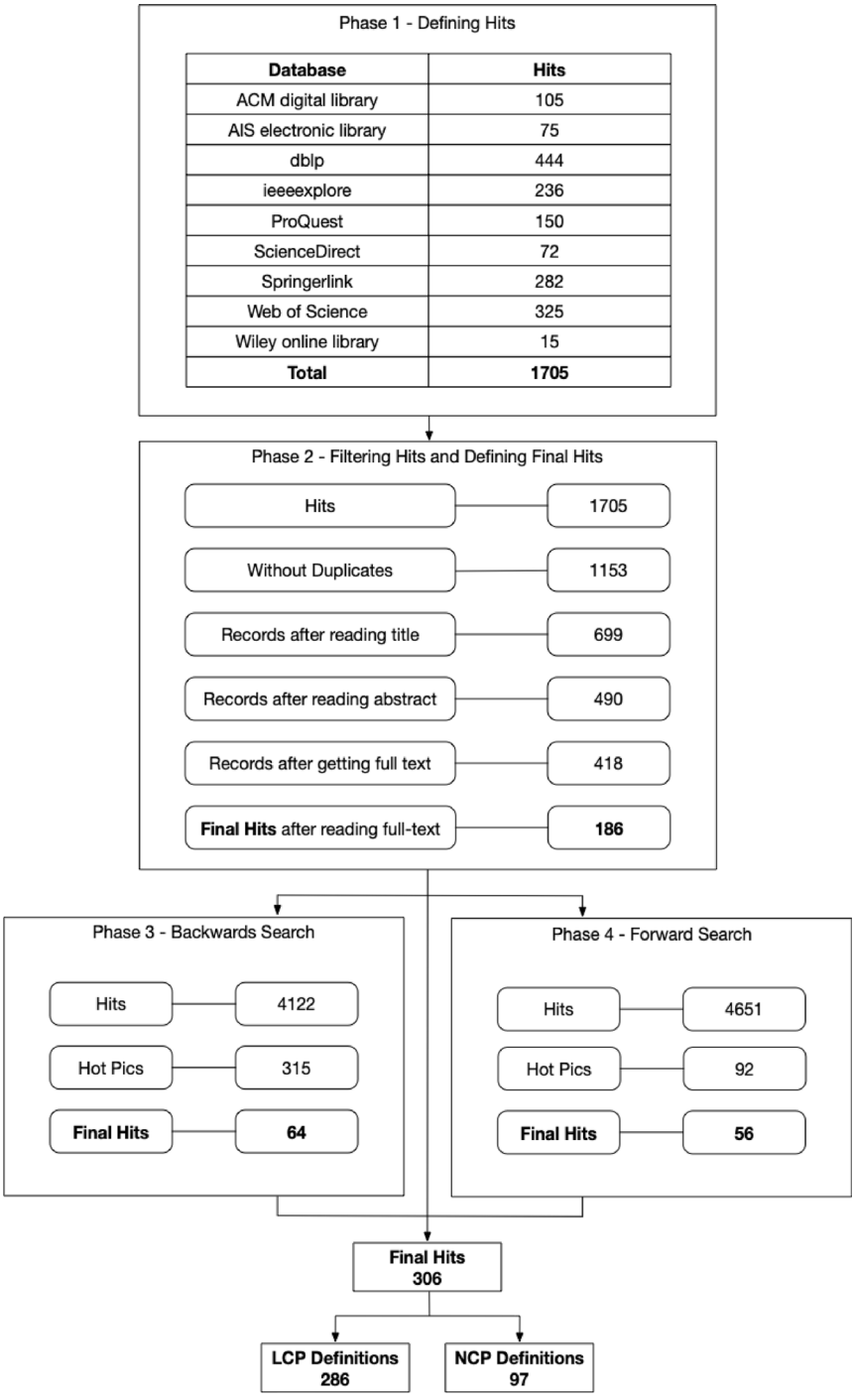

Fig. 3 Seach Process

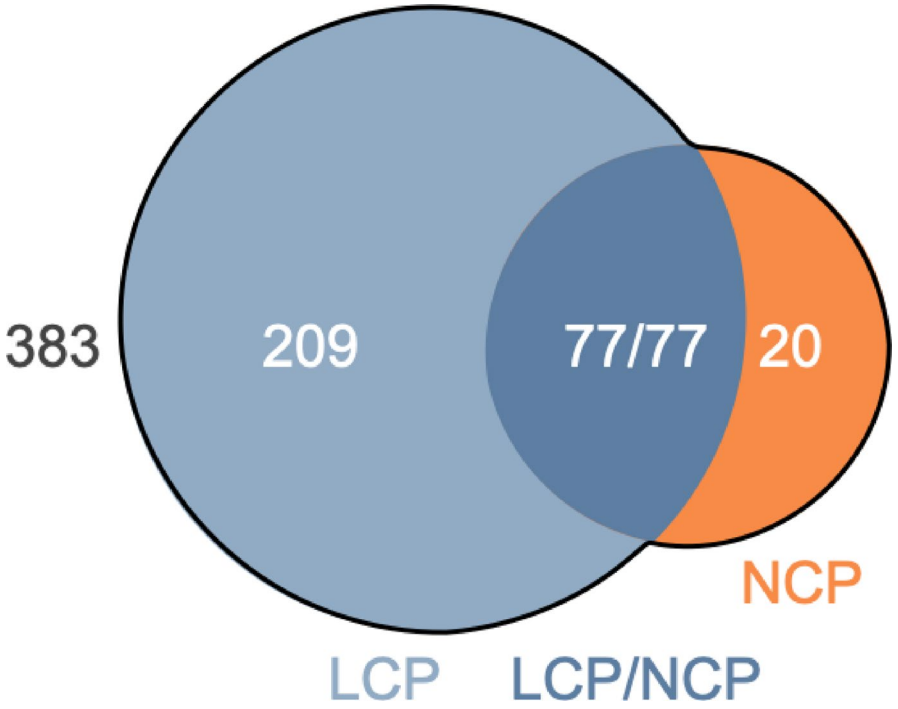

Fig. 4 Distribution of LCP and NCP definitions from Articles

tracing the relationships between definitions and identifying frequently cited source definitions.

3.5 Analysis procedure

The collected definitions and descriptions systematically following the definition analysis of Girard and Girard (2015) and Ritterbusch and Teichmann (2023) were examined and then synthesized using word frequencies to answer RQs.

For this purpose, the text parsing library spaCy (Explosion AI GmbH et al. 2025) for the Python programming language was used. In the first step, each definition was tokenized into individual words. In the second step, all special characters were removed. In the third step, the spaCy's standard stop word list was used to filter out common words that do not contribute meaningfully to the content. Additionally, these domain-specific terms "low-code", "low code", "no-code, " and "no code" were also removed as they are intrinsic to the topic and would otherwise dominate the results. In the fourth step, all remaining words were lemmatized, reducing them to their root forms to allow for consistent comparison. In the fifth and final step, duplicate words within each definition were removed so that each word would be counted only once per definition.

Importantly, this process was conducted separately for LCP and NCP definitions, resulting in two distinct word lists for analysis. In addition to the base frequency

analysis, the paper also experimented with weighting methods inspired by Ritterbusch and Teichmann (2023). Specifically, different weights based on the recency of the publication year, the type of definition (self-made or extended), and the number of mentions a definition received within the corpus was applied. However, the results of the weighted analysis closely mirrored those of the unweighted word counts. Therefore, it was decided to proceed with the simpler, more transparent method based solely on absolute word frequency.

4 Which scientific and science related definitions of the terms LCP and NCP do exist (RQ1)

The timeframe for the analysis was defined based on the evolution of terminology. Although an earlier source from Mendix around 2010 still referred to the concept as MDD (Henkel & Stirna 2010), the terms LCP were first coined by Forrester in 2014 (Richardson & Rymer 2014). Consequently, this SLR prioritizes the period from 2014 to December 2025 to capture the era in which LCP and NCP became established both as academic constructs and management paradigms. All analyzed definitions of LCP and NCP are available in the supplementary material 1.

The final corpus yielded a total of 306 unique definitions and descriptions. The volume of definitions remained marginal until 2019, followed by a period of growth (Fig. 5).

When examining the publication dates of LCP and NCP definitions, the earliest definitions of LCP appeared in 2010, while the first definitions of NCP emerged in 2019.

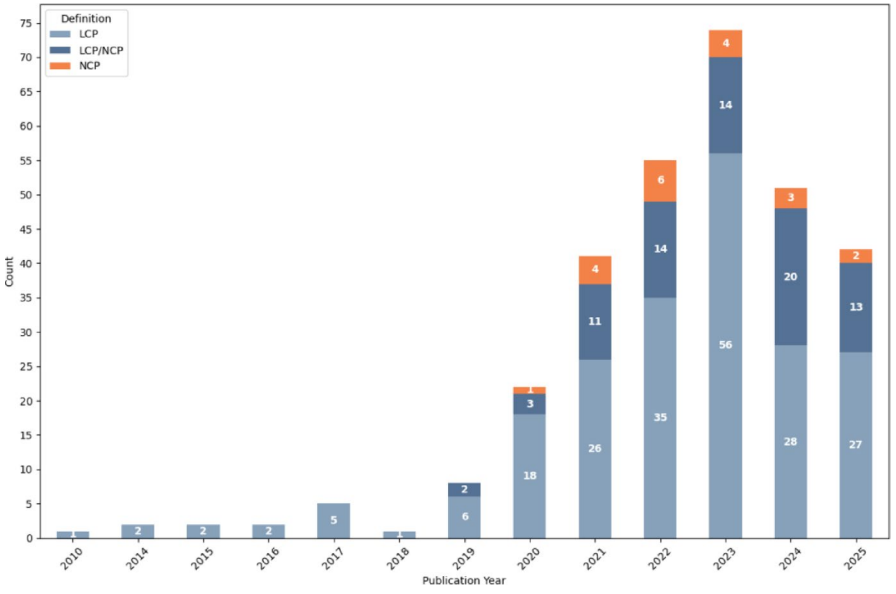

Fig. 5 Yearly distribution of final articles grouped by Definition

LCP definitions represent the primary focus of the literature, accounting for approximately 75% (n=286 out of 383) of the total extracted material. In contrast, NCP definitions constitute 25% (n=97) of the overall definitions.

4.1 LCP definitions

Of the 286 LCP definitions, 28,7% (n=82) were original conceptualizations, while 71,3% (n=204) were extended definitions. This high ratio of extension suggests a maturing consensus where authors build upon established definitions. The most definitions emerged in the year 2023 with 70 definitions and 2022 with 49 definitions (Fig. 6).

4.2 NCP definitions

For the 97 NCP definitions, the split was more balanced: 46,4% (n=45) were original conceptualizations and 53,6% (n=52) were extensions (Fig. 7). Most definitions were published in 2024 with 23 definitions and 2022 with 20 definitions.

The most frequently used definitions are presented in the next chapter, based on an analysis of their temporal distribution and citation frequency within the pool of 306 articles and its corresponding 383 definitions. All analyzed definitions of LCP and NCP are available in the supplementary material.

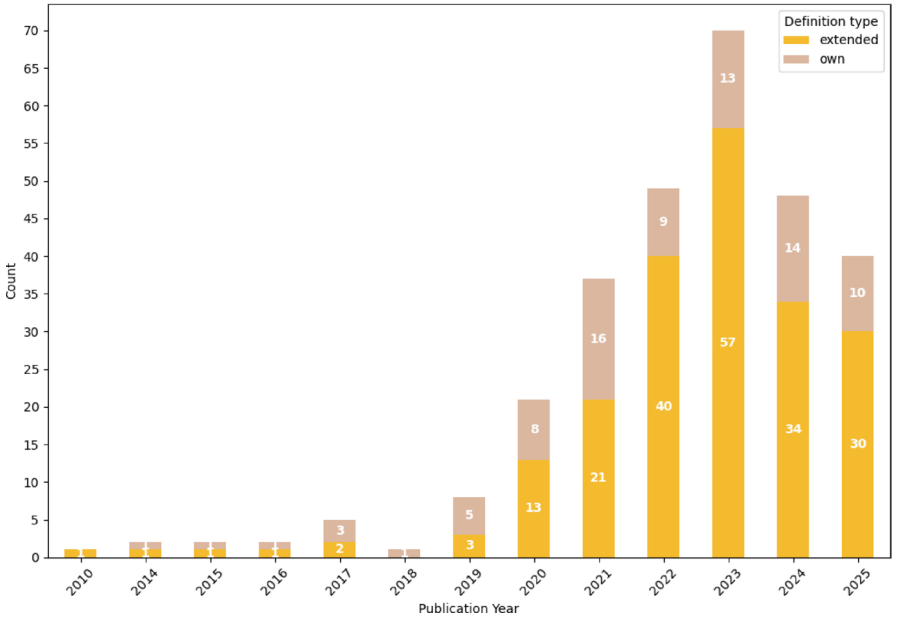

Fig. 6 Yearly distribution of own and extended LCP definitions

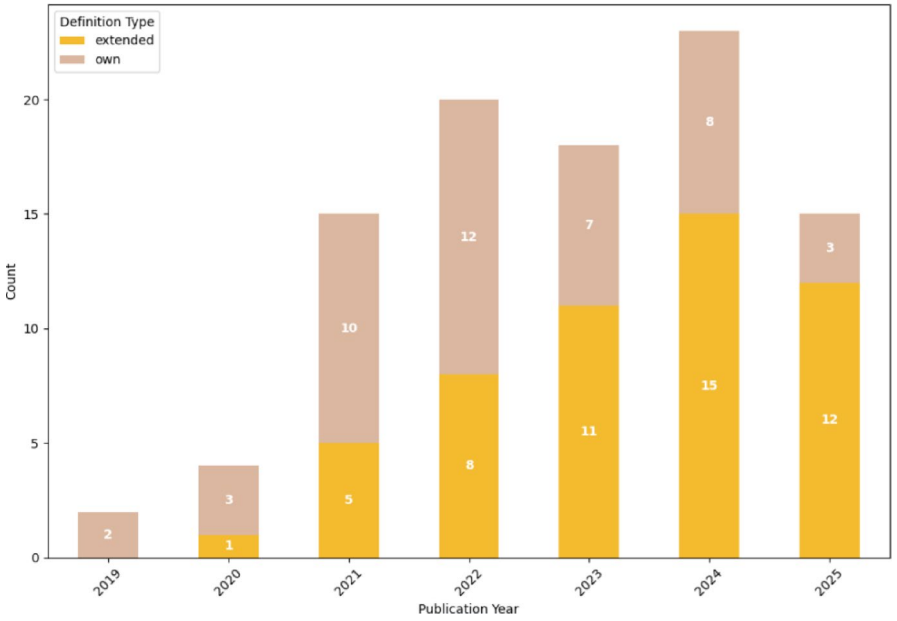

Fig. 7 Yearly distribution of own and extended NCP definitions

5 How does LCP/NCP definitions change over time (RQ2)

To examine the evolution of definitions for LCP/NCP, two distinct timeframes were selected: before 2014–2019 and 2020–2025. These periods were chosen based on significant developments in the conceptualization and market traction of LCP/NCP. The first timeframe, before 2014–2019, corresponds to the dawn of LCP terminology, coined around 2014, and captures the emergence of early definitions. The second timeframe, 2020–2025, reflects a period of rapid growth and broader adoption, where definitions expanded and diversified alongside technological advances and increasing industry impact. For each timeframe, up to the three most frequently cited references in the network graph within the collected literature were identified and analyze (Table 1).

5.1 LCP definitions before 2014 to 2019

The early framing of LCPs focused on the shift from manual, code-centric programming to visually driven application creation. Waszkowski (2019), Richardson and Rymer (2014) and Tisi et al. (2019) emphasized how LCPs enable RAD through graphical user interfaces rather than traditional hand-coded techniques. Primary functionalities were listed as support for databases, business processes, and user interface design, grounded in concepts of 4GL and RAD methodologies, with an explicit reduction in time spent troubleshooting code. The definition also situated LCP as a novel programming approach introduced around 2011, marking its emergence as a new paradigm (Waszkowski 2019).

Table 1 Timeframes and most cited LCP/NCP sources from network analysis

Type	Timeframe	References	# Network references
LCP	Before 2014 to 2019	Waszkowski (2019)	28
		Richardson and Rymer (2014)	27
		Tisi et al. (2019)	19
	From 2020 to 2025	Sahay, Indamutsa, et al. (2020)	44
		Di Ruscio et al. (2022)	25
		Sanchis et al. (2020)	25
NCP	Before 2014 to 2019	Vincent et al. (2019)	13
		Hyun (2019)	5
	From 2020 to 2025	Hurlburt (2021)	5
		Woo (2020)	4
		Lethbridge (2021)	3

Tisi et al. (2019) in this period described LCPs as cloud-based PaaS systems, which allowed users to build complete applications through dynamic graphical user interfaces, visual diagrams, and declarative languages. The role of citizen developers was acknowledged, though professional developers still formed part of the target group. LCPs were described for their suitability to domain-specific application types, such as databases, mobile, process automation, and request management, and anticipated growth in fields like IoT.

Richardson and Rymer (2014) focused on the promise of rapid delivery and minimal hand-coding, necessary for systems of engagement, with attention to setup and deployment speed.

5.2 LCP definitions from 2020 to 2025

In the later timeframe, definitions reveal a marked increase in theoretical maturity and architectural detail. Sahay, Indamutsa, et al. (2020) is describing LCPs as cloud-native software platforms supporting developers across diverse domain knowledge and expertise levels, with ready-for-production deployment as a core outcome. The architecture is typically multi-layered, integrating graphical modeling, data and service connections, deployment infrastructures, and back-end components. They build on model-driven engineering, automatic code generation, declarative languages, and cloud infrastructures to promote automation and abstraction. Key features include graphical interfaces, interoperability with external systems, security, collaborative and agile-oriented development, reusable templates, and scalability. Collectively, these enable the structured design and deployment of business logic (Lieberman et al. 2006).

Sanchis et al. (2020) also anchor LCPs in enterprise agility, stating that firms adopt low-code for rapid, test-and-learn delivery, leveraging prefigured components and visual interfaces to minimize engineering complexity and cost while accelerating response to market and internal needs.

Beyond these platform-centric perspectives, Di Ruscio et al. (2022) explicitly position LCP within the broader landscape of model-driven engineering. They concep-

tualize LCP as cloud-based environments that use visual and declarative techniques to minimize manual coding, integrate lifecycle management (LCM) and deployment, and thus make development accessible to both technical and non-technical users.

5.3 Emerge of NCP

The conceptualization of NCP has experienced notable development over time. Only some cited definitions from the periods before 2014 until 2019 exist then an emerged happened from 2020 to 2025 highlight shifts in focus from technical underpinnings to user experience and application range.

5.4 NCP Definitions before 2014 to 2019

In the period before 2020, NCPs were not widely conceptualized as independent categories. The prevailing view, as found in Gartner's definition (Vincent et al. 2019), considered NCPs as a subset or marketing expression within the broader LCP market. Under this model, the technical foundation for both LCP and NCP was visual modeling and configuration, the core difference lying in user exposure to code rather than platform capabilities. LCP were understood as runtime environments for application logic, incorporating LCM and enterprise-grade characteristics such as reliability, scalability, security, and integration with APIs and cloud services. Vincent et al. (2019) described LCPs (including NCPs) as systems that facilitate rapid development and one-step deployment via declarative, high-level abstractions, supporting user interfaces, business logic, and data services, but cautioning that improvements in productivity often came at the expense of portability and vendor neutrality.

A second contemporary definition (Hyun 2019) characterized NCPs as specialized LCP clouds leveraging special visual components for industry-, business-, or branding-specific functions. The main distinction highlighted was the required skill level: no-code enabled business users to build applications via visual methods, whereas LCP still required developers for backend integration and compatibility with other enterprise systems.

5.5 NCP definitions from 2020 to 2025

By 2020, the concept of NCPs had evolved, with a surge in both academic and commercial interest. Leading definitions now emphasized user empowerment, ease of use, and the democratization of application development.

Woo (2020) framed NCPs as environments enabling users to create web and mobile applications by dragging and dropping elements, eliminating the need for traditional coding. This definition expanded the historical context, positioning no-code development as the natural continuation of programming abstractions, from assembly to spreadsheets and What You See Is What You Get (WYSIWYG) design. Newer platforms began to integrate AI features, providing intelligent assistants and recommendations, marking a substantial shift in the platform's intelligence and user-centric design.

Lethbridge (2021) further broadened the scope, describing NCPs as platforms that include a wide spectrum of configurable business applications, ranging from content management and e-commerce to spreadsheet-based analytics and automation. The core functionality was described as graphical and form-based user interfaces, which allowed users to select, connect, and configure elements from both native and third-party libraries. This version of NCP moved definitively beyond mere IT enablement, cementing the term's association with broad business utility for end-users.

Hurlburt (2021) shows the distinction: NCPs permit citizen developers to build enterprise Software as a Service (SaaS) application with little or no programming skills, typically through WYSIWYG environments designed for process automation. The emphasis moved decisively away from enterprise IT concerns toward process usability, breadth of user adoption, and accessibility.

6 Defining LCP and NCP (RQ3)

After collecting and preparing all definitions, the next step focuses on analyzing their content to uncover shared conceptual patterns. This chapter therefore presents the linguistic and semantic structure of the collected definitions of LCP and NCP. The aim is to identify the most common terms, explore how they relate to one another, and derive a consolidated definition that reflects the core characteristics found across the literature.

6.1 Defining LCP definition

The following excerpt (Table 2), which includes the 30 most frequently occurring words for LCP, highlights terms that appeared at least 79 times across the analyzed definitions. Based on this set, a simplified and holistic definition can be constructed by integrating all terms that meet this frequency threshold. To improve clarity and avoid redundancy, semantically similar terms were consolidated. In this context,

Table 2 Definitions on the term LCP

Word	Quantity	Word	Quantity
Development	245	Model	104
Application	234	Tool	103
Platform	225	Drag	103
Software	179	Program	100
User	176	Component	99
Use	173	Provide	99
Visual	137	Drop	98
Interface	136	Allow	97
Developer	132	Programming	94
Business	128	Build	92
Code	126	Approach	90
Process	114	Environment	86
Create	111	Graphical	86
Develop	110	Base	85
Enable	105	Language	77

"build" represents both "build" and "create", "enable" stands in for both "enable" and "allow", and "visual" is used to encompass both "visual" and "graphical".

To gain a deeper understanding of how specific terms are used in relation to one another, all identified words were examined within the context of their original definitions. The aim was to interpret not just their frequency, but also their semantic usage and associative meaning. For example, the term "user" frequently appeared in the phrase "user interface", while "developer" was often used in reference to "citizen developers" and professional developers. Taking these contextual relationships into account, the following definition was derived:

Low-Code platforms are software application development platforms that enable professional and citizen developers to build business tools through visual user interfaces with minimal coding. These platforms use process-oriented development by providing model-based approaches and environments, drag-and-drop functionality and reusable components without the need for programming languages.

Sahay, Indamutsa, et al. (2020) was the most frequently referenced source for extended definitions in the network analysis, appearing 44 times within the pool of 286 LCP definitions, followed by Waszkowski (2019) with 28 times.

Sahay, Indamutsa, et al. (2020) describes LCP as cloud-based environments that enable developers with varying levels of technical expertise to build complete applications. Their definition emphasizes MDE, automatic code generation, and the use of declarative and high-level abstractions. Their framework is structured into four layers which are grouped and three tiers. The layers are the application layer including a toolbox and widgets, the service integration layer for external services, the data integration layer linking data like databases and spreadsheets, and the deployment layer for cloud or local infrastructures. On the server side, code generation, compilation, optimization, and services manage business logic, data models, and security rules to support efficient application delivery. The tiers are grouped into application modeler, the server side and integrating of external services.

In contrast, the definition derived in this study also highlights the dual user group of professional and citizen developers, the use of visual interfaces, model-based approaches, and reusable components. However, the definition from Sahay, Indamutsa, et al. (2020) places more emphasis on the architectural features of a LCP. Therefore, while both definitions share key ideas such as visual development and model-based abstraction, the definition by Sahay, Indamutsa, et al. (2020) provides a more technical and structural perspective, whereas the definition derived from the analyzed corpus offers a more user-centric and functional view of LCP characteristics.

Waszkowski (2019) defines LCP as tools that rely on graphical user interfaces instead of traditional hard-coded programming. They focus on developing databases, business processes, and web-based user interfaces. Grounded in the principles of 4GL and RAD, LCPs allow developers to concentrate more on design and functionality than on syntax. This reduces development time and simplifies troubleshooting.

While Waszkowski (2019) emphasizes the roots of LCP in 4GL and RAD, with a strong focus on graphical user interfaces for designing databases, business processes, and web-based frontends, the definition derived in this study takes a broader perspective. It highlights the dual role of citizen and professional developers, stresses visual user interfaces, and integrates process-oriented, model-based approaches with

reusable components, thereby reflecting a more comprehensive and practice-oriented understanding of LCP.

In conclusion, the analysis shows that LCP definitions consistently emphasize visual development, model-based approaches, and minimal coding to support both professional and citizen developers. While Sahay, Indamutsa, et al. (2020) focus on the architectural and technical foundations of LCP, and Waszkowski (2019) highlights its process-oriented and design aspects, the definition derived in this study combines these perspectives. It captures both the technical foundation and the user-oriented character of LCP, positioning it as a bridge between professional software engineering and accessible application development.

6.2 Defining NCP definition

The following excerpt (Table 3), consisting of the 30 most frequently occurring words for NCP, presents terms that appeared at least eight times in the analyzed definitions. Based on this set, a simplified and holistic definition can be derived by incorporating all terms that meet this frequency threshold. To ensure clarity and avoid redundancy, semantically similar terms were consolidated. Specifically, "develop" is used as a representative for "develop", "build" and "create", "enable" serves as a proxy for both "enable" and "allow", and "visual" encompasses both "visual" and "graphical".

To gain a deeper understanding of how specific terms are used in relation to one another, all identified words were examined within the context of their original definitions. The aim was to interpret not just their frequency, but also their semantic usage and associative meaning. For example, the term "user" frequently appeared in the phrase "user interface", while "code" was often used in reference to eliminate coding. Taking these contextual relationships into account, the following definition was derived:

No-code platforms are software application developments platforms that enable users to develop business tools through visual user interfaces, without writing code or requiring programming knowledge like a developer. These platforms provide drag-

Table 3 Definitions on the term NCP

Word	Quantity	Word	Quantity
Application	59	Drag	28
Development	55	Business	28
Platform	48	Require	28
User	45	Enable	27
Tool	36	Drop	26
Use	34	Build	26
Software	34	Knowledge	26
Code	34	Provide	25
Developer	33	Develop	24
Without	32	Component	22
Programming	32	Approach	22
Visual	31	Environment	22
Interface	30	Allow	20
Create	30	Process	20
Program	29	Base	20

and-drop approaches, reusable components and process environments, eliminating the need for code-based programming.

The most frequently cited extended definition in the network analysis was Vincent et al. (2019) referenced in nine instance in the pool of 65 definitions followed by Woo (2020) with seven times.

Vincent et al. (2019) describe NCP as a marketing term rather than a separate technology. They see NCP as a subset of LCP enabled by visual modeling and typically aimed at non-developers or citizen developers. Technically, NCP shares the same foundations as LCP, including model-driven development and LCM. The key distinction lies in user focus and abstraction level, not in core functionality.

Woo (2020) frames NCP and LCP as a natural evolution of programming, continuing the trend toward more intuitive development seen since the shift from assembly to high-level languages. It highlights the use of drag-and-drop interfaces and predefined components as core features. Moreover, it notes that some platforms now integrate AI to provide data-driven suggestions, making app creation even more accessible and intelligent.

The analysis of NCP definitions highlights their focus on accessibility, automation, and visual development. The results show that NCPs empower non-technical users to create business applications through intuitive interfaces and reusable components. In line with Vincent et al. (2019) and Woo (2020), NCP emerges as an evolution of LCP distinguished by a higher level of abstraction that eliminates the need for coding.

7 Comparing LCP and NCP definitions (RQ4)

When comparing LCP and NCP based on the derived definitions, several similarities and differences become apparent.

Regarding similarities, both LCP and NCP can be classified as software application development platforms. They incorporate drag-and-drop functionality, which supports intuitive interaction during the development process. Furthermore, they rely on visual user interfaces that abstract away much of the technical complexity of traditional programming. Both approaches also make use of reusable components and component libraries, which increase efficiency and modularity in application creation. Finally, they share the characteristic of enabling new groups to engage in development: while NCPs provide enablement for users without technical expertise, LCPs extend this possibility to both citizen developers and professional developers.

In contrast, several differences can be identified. With respect to the user group, NCPs are directed primarily at end-users without programming knowledge, whereas LCPs explicitly target citizen developers as well as professional developers. While both incorporate process orientation, LCPs emphasize explicit model-based approaches and structured development environments for complex business process design, whereas NCPs provide abstracted process support through predefined components and templates.

7.1 Comparison of literature perspectives for LCP and NCP differences

Furthermore, during the data collection phase, differences in LCP and NCP were collected from papers, if these were directly mentioned. Beyond the comparison derived from the definitions, the literature reveals further differences in how LCP and NCP are conceptualized.

For instance, Curty et al. (2022) and Avishahar-Zeira and Lorenz (2023) position NCP as a subset of LCP. In this perspective, NCP represents a more abstract level where code is entirely hidden from the user, whereas LCP requires minimal but still some programming knowledge. This view stresses continuity rather than a strict dichotomy between the two approaches.

At the same time, the literature points to a certain ambiguity in terminology. Di Ruscio et al. (2022) notes that while the term NCP is widely used in marketing contexts, some market analysis firms oppose using it as a distinct category like Gartner coining this (Vincent et al. 2019). Instead, LCP and NCP are often treated interchangeably, and in some contributions, only the term LCP is used. This reflects that the distinction between LCP and NCP is not only technical but also discursive, shaped by the market context and research perspective in which the platforms are examined.

Other contributions argue for a clearer differentiation. According to Picek (2023) and da Cruz et al. (2021) LCP require at least a certain level of programming knowledge and are therefore intended for developers, while NCP remove coding altogether and can thus be used by everyone. Similarly, Favre et al. (2023) emphasizes that NCP embody the principle of simplicity over expressiveness, offering fewer features to ensure usability for lay persons. In contrast, LCP focus on enhancing functionality and extensibility for more advanced users.

Further distinctions are drawn regarding technical capabilities. DeSilva, et al. (2023) highlights that both approaches share visually integrated development environments and support for business application creation. However, LCP supports application integration and customization, while NCP does not. As a result, NCP is associated with lower costs, simplicity of use, and limited customization, whereas LCP is connected to higher costs, greater complexity, and extensive customization possibilities. Dobler et al. (2022) and Abdul Razak et al. (2024) also underline that LCP enables the modeling of more complex solutions, whereas NCP remains constrained to simpler implementations.

8 Discussion and future work

8.1 Summary of findings

8.1.1 Technological evolution of LCP and NCP: from MDE to AI integration

This SLR analyzed the conceptual evolution and linguistic foundations of LCP and NCP from 2014 through end of 2025. Our analysis reveals that these platforms are not isolated technological phenomena but are deeply grounded in the historical para-

digms of MDE. By abstracting requirements into domain models and automating code generation, LCPs and NCPs enable a fundamental shift in software engineering: moving the focus from technical implementation to business logic (Cabot 2020; Hailpern & Tarr 2006). Beyond their MDE foundations, the integration of visual programming (Jost et al. 2014) and RAD principles marks a pivotal moment in the democratization of IT. By reducing the reliance on extensive syntax-based programming expertise, these paradigms have lowered the barrier to entry for citizen developers by reducing the need for extensive programming expertise an making shorter development cycles (Beynon-Davies & Williams 2003). The significance of this evolution lies in the convergence of development accessibility and Cloud Infrastructure. As LCM and maintenance move to centralized, cloud-based environments, the traditional friction between business needs and IT delivery is minimized (Di Ruscio et al., 2022). Centralized deployment and management through the cloud facilitate continuous updates, easier scalability, and simplified administration. This shift does not merely improve platform usability, it fundamentally redefines enterprise agility.

Beyond these foundational technologies, emerging developments such as AI are further extending the functional scope of LCP and NCP. LLMs assist users by generating interface elements and enabling natural-language queries that access underlying databases through the model context protocol (MCP). This capability allows users to retrieve analytical summaries or reports directly in natural language (Jiang & Hu 2025). New interaction paradigms such as vibe coding complement rather than replace LCP and NCP, as they introduce conversational and AI-assisted development that can be seamlessly integrated into existing LCM frameworks (Gadde 2025). Owing to their mature LCM structures, LCP and NCP are well positioned to incorporate vibe coding functionalities while ensuring that applications remain maintainable and controllable within enterprise environments.

The integration of agentic AI represents a fundamental shift from instruction-based development to autonomous decisioning. These AI agents operate within adaptive systems to handle complex, manual tasks, significantly enhancing the quality of decision support and process automation (Jiang & Hu 2025). This transition is epitomized by the emergence of vibe coding and natural-language prompting, which reposition the development focus from manual UI design to conversational intent. Ultimately, the fusion of AI with LCP/NCP environments enables systems to learn and reason beyond predefined rules, transforming platforms from simple execution engines into intelligent, autonomous agents of enterprise automation (Viswanadhapalli 2025).

8.1.2 Conceptual emergence and institutionalization of LCP and NCP

The analysis of sources reveals that the term low-code became widely established primarily through the influence of research and consulting firms such as Gartner and Forrester (Di Ruscio et al., 2022). These institutions played a role in disseminating the terminology through market studies. However, it is evident that many companies had already developed platforms and tools that embodied the principles of today's LCP and NCP long before the terminology was introduced (Bock & Frank 2021). The classification and conceptual framing by the consulting firms subsequently provided a unified vocabulary that shaped both market perception and research discourse.

As a result, the topic gained sharper definition, leading to the emergence of numerous start-ups and platform developments that have since broadened the diversity of the ecosystem and accelerated technological innovation in this field (Bock & Frank 2021; Di Ruscio et al., 2022).

8.1.3 Defining LCP and NCP

The core of our findings is the derived differentiation between LCP and NCP based on their semantic usage in the literature. NCPs emerge as highly abstracted software environments to develop business applications via visual interfaces designed to democratize development for non-technical citizen developers (Abendroth & Bender 2025) by eliminating manual coding entirely through drag-and-drop mechanisms and predefined components. Because NCP are tailored to users without programming experience, they depend heavily on predefined components, connectors, and templates to facilitate rapid onboarding. Consequently, providers must emphasize comprehensive documentation, guided training, and accessible process libraries or marketplaces to support effective use (Abendroth & Bender 2025).

In contrast, LCP are designed to support both professional and citizen developers. They enable application creation through visual modeling while allowing extensions using conventional programming languages. LCP combine model-based development, process orientation, and integrated environments that balance visual design with the flexibility of manual coding. This definition is consistent with the majority of academic and industry sources, which characterize LCP as hybrid systems bridging traditional programming and visual modeling (Cabot 2020; Hailpern & Tarr 2006). Unlike NCP, LCP target heterogeneous user groups with varying technical expertise. Professional developers use them to accelerate development and standardize architectures, while non-professional users benefit from abstraction mechanisms that lower technical barriers.

Conversely, LCPs act as hybrid systems that bridge traditional programming and visual modeling, targeting a heterogeneous user group that includes both professional developers seeking architectural standardization and business users requiring extensibility (Cabot 2020). Rather than strictly separate categories, the analysis suggests these platforms exist on a continuum of accessibility and extensibility. This distinction is further complicated by market specializations, such as Business Process Automation (BPA) and Integration Platforms as a Service (iPaaS), which vary significantly in their technological focus and the complexity of tasks they perform (Baumgarten et al. 2025). Historically, the field experienced a gradual initial ascent in conceptual interest, which has recently transitioned into a period of rapid exponential growth and institutionalization.

8.1.4 Market and user role differentiation of LCP and NCP

The distinction between LCP and NCP becomes particularly evident when considering user roles and their interaction with programming languages. While NCP are primarily oriented toward business users who do not engage in coding, LCP provide an optional coding layer that enables experienced users to implement customized logic

or integrate external systems. This structural difference reflects the dual objective of LCP to promote both accessibility and extensibility. Consequently, LCP and NCP should be viewed as points on a continuum rather than as strictly separate categories, since many commercial solutions combine features of both approaches.

The analysis further reveals that both ecosystems exhibit multiple market specializations. These specializations can be differentiated according to their technological and functional focus, such as data processing, business and process logic, or system integration (Baumgarten et al. 2025). At the same time, LCP and NCP differ considerably in their target audiences. NCP primarily address business users and non-experts seeking to automate simple workflows, whereas LCP target a broader range of users, including professional developers and consultants who manage more complex use cases.

Recent research also highlights several user roles within these ecosystems (Baumgarten et al. 2025). These include non-experts such as citizen developers and low-code champions who focus on simple automation and tool creation, and low-code experts such as platform consultants and professional developers who operate at higher levels of abstraction and integration. These roles vary not only in technical proficiency but also in the complexity of tasks performed. Accordingly, platform suitability depends strongly on the intended user group and the required level of sophistication. Advanced platforms offer extensive integration and customization options for expert users, while simpler platforms emphasize usability and guided interaction for business users.

Moreover, a standardized measurement of user capability levels remains absent. Although some studies propose frameworks to assess application complexity (Abendroth & Bender 2024), no established method exists to quantify user proficiency across roles and platform types. Developing such a framework would be valuable for both research and practice, particularly in defining job profiles in process and digitalization management where role clarity and skill differentiation are essential.

8.2 Functional categorization and taxonomy alignment

In analyzing the diverse landscape of LCP and NCP, a clear evolutionary trajectory toward functional specialization emerges. A fundamental distinction must be drawn between incumbent system categories that are now integrating low-code layers into their existing architecture and standalone LCP/NCP that are increasingly incorporating features traditionally found in specialized system classes.

This leads to a classification of systems utilizing a low-code-based approach across several established system classes (Baumgarten et al. 2025; Noack & Fedtke, 2025):

- *Standalone LCP and NCP*: Platforms embodying this study's primary definitions, utilizing visual, model-based, and process-oriented environments to enable development with minimal (LCP) or no (NCP) traditional coding, represented by LCPs such as SCOPELAND or Retool, and NCPs like Bubble or Unqork (Pöppler & Abendroth 2025).
- *Integrated LCP in ERP systems*: Established ERP systems, which utilize low-code to allow for agile extensions of the core ERP logic, such as SAP Build,

Oracle Fusion or engomo (Abendroth et al. 2026).

- *Multi-experience development platforms (MXDP)*: MXDPs like Mendix prioritize a the user experience by harmonizing interactions across a wide array of digital channels and devices. These platforms enable the parallel development of applications for web and mobile endpoints, as well as emerging channels such as chat apps, smartwatches, and immersive environments like Augmented or Virtual Reality. While low-code-oriented MXDPs include connectors for third-party systems and data persistence, these integration aspects are generally less emphasized than the core focus on multi-channel consistency. This category encompasses modern *CRMs* like Salesforce, which utilize extensive low-code capabilities to extend customer-facing workflows across various devices, and *Groupware* such as HCL Notes integrated with HCL Volt MX, which has evolved to support rapid, multi-device application development within collaborative workspaces.
- *Business process automation (BPA):* Lightweight automation tools such as n8n, Zapier, and IFTTT, which focus on connecting disparate services via logic-driven triggers.
- *Business process management suites (BPMS)*: Robust orchestration environments like Bizagi or Camunda that emphasize end-to-end process modeling and execution.
- *Robotic process automation (RPA)*: UI-driven automation tools, including UiPath or SS&C Blue Prism, which bridge legacy gaps through bot interactions.
- *Document management system (DMS)*: Document-centric systems such as Nutrient, which provide low-code capabilities for automating complex document lifecycles, processing, and workflows.
- iPaaS (Integration Platform as a Service): Specialized integration hubs such as SEEBURGER or Lobster_data. These systems utilize low-code visual mapping and configuration to manage complex EDI, API management, and enterprise application integration.

To support this categorization, a comprehensive list of 76 tools classified into 10 system classes is provided including LCP/NCP in the supplementary material 2, offering a view of the current market diversity and the technological overlap between standalone platforms and integrated enterprise solutions.

To further contextualize the functional diversity of these systems, the following taxonomy maps the identified classes across two primary dimensions: the functional perspective and the design options for the user interface. This framework, synthesized from the works of Baumgarten et al. (2025) and Noack & Fedtke (2025), illustrates how platforms are positioned based on their operational focus, ranging from data processing to system integration and the complexity of their frontend capabilities (Fig. 8).

As shown in the classification, systems such as iPaaS and DMS typically occupy the secondary functional perspective, as they focus primarily on more easy to use headless background automations, data orchestration, and rule-based decision logic rather than complex user-facing designs. In contrast, LCP and NCP are characterized as general-purpose environments providing the most advanced options for individual UI design and sophisticated business logic. They differ primarily in that NCPs priori-

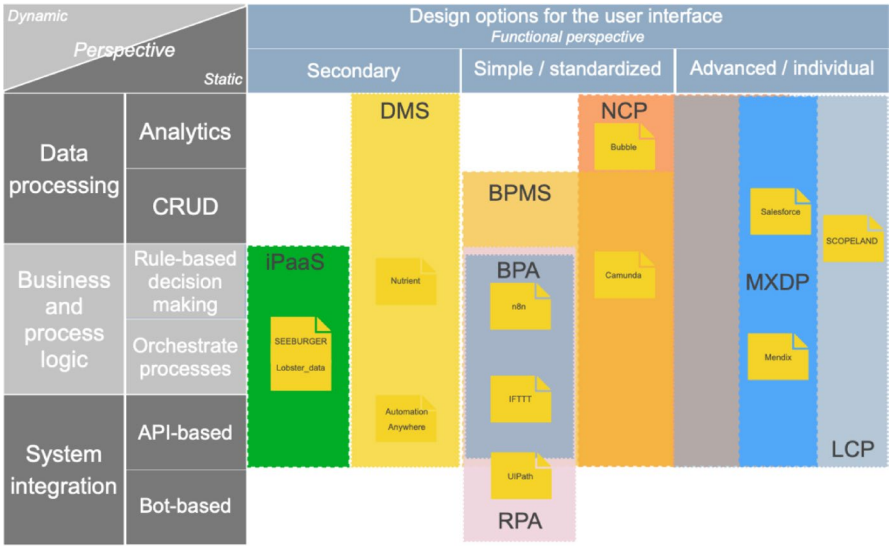

Fig. 8 Categorization of common Low-code based products based on (Baumgarten et al. 2025; Noack & Fedtke, 2025)

tize ease of use by abstracting away manual coding, whereas LCPs offer extensibility through the integration of custom code. Between these poles, BPMS prioritize the digitalization of process maps, workflows and decision rule automations, often utilizing standards like BPMN, while BPA only focuses automations. RPA focuses on the bottom layer of system integration, utilizing bots to automate repetitive tasks via existing user interfaces when defined APIs are unavailable. Expanding on the general-purpose tier, MXDP emerge as a specialized subgroup of LCP. These platforms harmonize the user experience across diverse digital touchpoints, extending the functional reach across the entire vertical spectrum, from system integration to advanced analytics, while ensuring UI consistency across web, mobile, wearables, and immersive environments.

Taken together, these findings show that LCP and NCP capabilities have diffused far beyond dedicated platforms into a wide range of incumbent system classes, while at the same time giving rise to new, highly specialized low-code based solutions.

8.3 Challenges and implications

The synthesis of the analyzed literature reveals three primary thematic pillars of challenges and implications: organizational, technical, and ecosystem-market dynamics. Table 4 summarizes the main challenges and issues identified from the basket of literature and offers recommendations and directions for future research.

8.3.1 Organizational implications

Governance in organizations: A fundamental organizational implication identified in this study is the shift toward decentralized development and its governance. The

Table 4 Research agenda

Main challenges	Issues	Research recommendation
Organizational implications	Governance in organizations	Determination of governance frameworks and role-based checklists to streamline enterprise-wide adoption and management of decentralized development
	Strategic skill development	Identification of a skill level maturity model to curate learning materials and quantify user proficiency, ensuring an optimal learning curve for various user roles
	Lowering entry barriers	Identification of specific mechanisms to further lower entry barriers for non-technical citizen developers, particularly within the context of NCP environments
Technical implications	Composable architecture and hybrid integration	Identification of plug-and-play strategies that leverage Microservices and API-first designs (REST/SOAP). Also, identify alternative integration options, where legacy systems lack native API support, by potentially utilizing wrapper technologies
	Open standards and the metamodel gap	Establishment and further development of open standards for metamodels to ensure cross-platform compatibility and mitigate the risks of vendor lock-in
	Modeling, customization, and semantic richness	Investigation of frameworks for metamodel customization, specifically regarding the import of BPMN 2.0 processes and the readiness of Domain-Specific Languages (DSL)
	iPaaS orchestration	Development of evaluation frameworks for iPaaS to help companies align these tools with their existing IT architectures
Ecosystem-market dynamics	User-vendor collaboration and platform co-evolution	Investigation of collaborative programs between users and vendors to influence the development of reusable platform features and components
	Ecosystem collaboration and conflict	Investigation of collaboration and conflicts potential between different stakeholders in the ecosystem (vendors, users, integrators, consulting companies)

identified trend toward decentralized development in our study suggests a fundamental shift in the role of corporate IT, from a sole provider of solutions to an enabler of innovation. This implies that research must focus on the determination of governance frameworks. Specifically, there is a theoretical and practical need for role-based checklists that clearly delineate responsibilities between central IT and business-unit citizen developers. Such frameworks are essential to streamline enterprise-wide adoption while ensuring security, compliance, and data integrity.

Strategic skill development: The effectiveness of LCPs is inherently contingent upon the heterogeneous capabilities of the citizen developer workforce (Bernsteiner et al. 2022; Matook et al. 2023). Our findings suggest a shift away from viewing users as a monolithic group, favoring instead the development of skill-level maturity models (Abendroth & Bender 2024). Research should theorize tiered proficiency frameworks that allow organizations to curate targeted learning paths and objectively quantify user competencies across diverse organizational roles (Elshan et al. 2025).

Lowering entry barriers: The recurring themes of no-code and visual modeling in our dataset highlight a strategic imperative to investigate the specific mechanisms that lower entry barriers for non-technical citizen developers. While LCPs are inherently more accessible than traditional coding, the question for organizations lies in identifying the threshold at which a business user transitions from a passive observer

to an active creator. Research should focus on the efficacy of NCP environments that utilize AI-assisted co-authoring, natural language processing, and logic-templates to bypass technical friction. This implies that the focus must shift toward usability engineering and cognitive load reduction.

8.3.2 Technical implications

Composable architecture and hybrid integration: Our findings imply a transition toward composable architecture, where LCPs act as plug-and-play components within the enterprise IT landscape. This requires researchers to look beyond the platforms themselves and investigate hybrid integration strategies, specifically how API-first designs (REST/SOAP) and microservices can be used to anchor LCPs into existing system landscapes, enabling a synergy between legacy stability and low-code agility (Viswanadhapalli 2025). Furthermore, research must identify alternative integration paths for legacy systems lacking native API support, specifically by evaluating the role of wrapper technologies or middleware adapters to bridge the gap between rigid back-end environments and agile, low-code front-ends.

Open standards and the metamodel gap: The prominence of model-based approaches in our linguistic analysis reveals a significant theoretical gap regarding the standardization of metamodels. The current prevalence of proprietary black-box models creates risks of vendor lock-in and hinders long-term application portability. Researchers should therefore prioritize the development of open, execution-oriented standards for metamodels and repositories to enable cross-platform compatibility. Projects such as the EU-funded Lowcomote demonstrate the potential for shared infrastructural backbones to reduce lock-in, facilitating a transition from simple LCPs to comprehensive Low-Code Engineering Platforms (Colantoni et al. 2020).

Modeling, customization, and semantic richness: The research community must move beyond general platform abstractions to investigate frameworks for metamodel customization. While our analysis identifies process-orientation as a core characteristic of LCPs, there remains a theoretical and practical disconnect between standard business process modeling and LCP execution. Researchers and vendors should specifically examine the importability of BPMN 2.0 processes into LCPs to ensure that business logic remains portable and grounded in industry standards (Alfonso et al. 2025). Beyond technical portability, vendors should provide standardized reference processes that serve as pre-configured blueprints, enabling customers to adopt industry-specific best practices more efficiently while maintaining architectural consistency. Furthermore, investigating the readiness of Domain-Specific Languages DSLs within LCPs is relevant for the discipline and needs to establish a standard (Bragança et al. 2021; Martínez-Lasaca et al. 2025). This shift transforms LCPs from generic app builders into semantically rich tools capable of reflecting sophisticated business domains.

iPaaS orchestration: The semantic analysis, which highlights data integration and service layers as core components, suggests that the theoretical potential of LCPs is currently throttled by ES. To overcome this, researchers must theorize the role of iPaaS as a centralized data hub for low-code ecosystems. This shift moves the focus from app-building to data orchestration, where iPaaS serves to aggregate and struc-

ture data from siloed and legacy ES into a consumable format for LCP/NCP (Sarabu 2025).

8.3.3 Ecosystem and market dynamics implications

User-vendor collaboration and platform co-evolution: The heavy emphasis on reusable components and template libraries within the linguistic analysis suggests that the value of an LCP/NCP is increasingly defined by its ecosystem rather than its core features alone (Abendroth & Bender 2025). This highlights a critical need for research into collaborative programs between users and vendors to influence the development of platform capabilities. Traditionally, software development followed a linear path where vendors dictated the feature set; however, the modular nature of LCP/NCP allows for a model of platform co-evolution (Gawer & Cusumano 2014). Researchers should investigate formal frameworks for lead-user integration, where enterprise IT departments and citizen developers actively contribute to the vendor's roadmap to ensure the development of functionally relevant, reusable components. This implies shifting the research focus from software procurement to ecosystem governance. By understanding the mechanisms that allow users to influence platform maturity, researchers can provide a blueprint for how organizations can move from being passive consumers of technology to active co-creators of the tools they rely on.

Ecosystem collaboration and conflict: Finally, our results point toward a significant need to theorize the multi-stakeholder dynamics of the LCP/NCP ecosystem. While current literature often focuses on the user-vendor dyad, the proliferation of these platforms introduces a complex web of interactions and potential conflicts between vendors, end-users, system integrators, and consulting firms (Abendroth & Bender 2025). As platforms move toward higher abstraction, traditional system integrators and consulting companies face a potential value-chain reconfiguration where their traditional coding services are commodified, leading to friction regarding service models and intellectual property. Conversely, collaboration is fostered through community-driven marketplaces and third-party plugin development. Research should therefore investigate frameworks for ecosystem governance that address these power asymmetries and misaligned incentives.

9 Conclusion and limitations

9.1 Implication for practice

For *companies* adopting or using LCP and NCP, it is essential not only to evaluate the initial implementation of these platforms but also to consider the strategic integration of emerging technologies within their digital transformation agendas. Organizations should continuously monitor the Return on Investment of LCP and NCP initiatives to ensure that these efforts lead to measurable efficiency gains, enhanced flexibility, and sustainable value creation. At the same time, companies need to establish governance frameworks that guarantee the consistency, maintainability, and scalability of applications developed across various departments and user groups.

Vendors of LCP and NCP should strengthen their understanding of the diverse user groups engaging with their platforms and adapt their offerings accordingly. This includes the provision of clear documentation, training programs, and support services tailored to both professional developers and citizen developers. Vendors should differentiate their product portfolios based on user expertise, domain requirements, and organizational maturity. Moreover, they should actively promote the integration of complementary technologies such as LLM-based assistants, agentic AI, and MCP architectures to expand platform functionality and improve user experience. Finally, vendors are encouraged to cultivate an open and sustainable ecosystem that facilitates component reusability and fosters collaboration among developers. Such an ecosystem should include accessible marketplaces, standardized documentation, and transparent communication channels between vendors and users (Abendroth & Bender 2025; Di Ruscio et al., 2022).

9.2 Theoretical contribution

The theoretical contribution of this study lies in systematically consolidating and conceptualizing the scientific foundations of LCP/NCP through a structured review and linguistic-semantic analysis. By bridging fragmented terminology and interpretations across more than a decade of publications, this research provides a unified theoretical framework that clarifies how the meanings of LCP and NCP have evolved and differentiated over time. The study contributes to theory building by (1) demarcating the conceptual boundaries between LCP and NCP based on user group, coding involvement, and process orientation; (2) revealing the temporal transition from technical and architectural emphases toward user-centric and democratization-oriented definitions; and (3) deriving consolidated and empirically grounded definitions for both LCP and NCP that integrate linguistic frequency and contextual meaning. By exposing the shifting semantics and usage contexts before 2014–2025, the article offers a stable conceptual foundation for future empirical and design-oriented research on visual development environments and digital democratization in software engineering.

Limitations: The rapidly evolving nature of LCP and NCP ecosystems presents a key limitation to this study. Because platform vendors continuously introduce new features, integration mechanisms, and pricing models, some findings may become outdated over time.

Future research should further investigate how LCP and NCP evolve within the broader framework of low-code management. Research should focus on understanding the required skill levels across different user groups, platform types, and market specializations. Such analyses would enable a clearer classification of user roles and help to determine how user capability and task complexity influence the quality and maintainability of developed applications. Moreover, future research should explore the integration potential of adjacent technologies, including LLM-based prompt engineering, vibe coding, and autonomous AI agents. These emerging paradigms could significantly enhance automation, collaboration, and knowledge transfer within LCP and NCP ecosystems, thereby enabling more intelligent, adaptive, and sustainable forms of application development and management.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11301-026-00603-2.

Acknowledgments Sincere thanks are due to Georg D. Ritterbusch for his methodological support in defining the conceptual framework of this study. The author also acknowledges Dr. Malte R. Teichmann for his diligent internal review of the manuscript. Their insights and feedback significantly contributed to the final quality of this work. Finally, appreciation is expressed to Prof. Dr.-Ing Norbert Gronau for his valuable insights during the research colloquia, as well as the entire team at the Chair of Business Informatics, especially Processes and Systems, for their collective support and for providing a stimulating environment throughout the duration of this study.

Author contributions The author confirms being the sole contributor to this work and has approved it for publication.

Funding Open Access funding enabled and organized by Projekt DEAL. The author has no relevant financial or non-financial interests to disclose.

Data availability All data supporting the findings of this study are available within the paper and its supplementary materials 1 and 2.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethics approval and consent to participate Ethical approval and consent to participate were not required for this study, as it did not involve human participants, identifiable personal data, or animals.

Consent for publication The author confirms being the sole contributor to this work and has approved it for publication.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Abdul Razak SF, Ernn Y, Yussoff FI, Bukar U, Yogarayan S (2024) Enhancing business efficiency through Low-Code/No-Code technology adoption: insights from an extended UTAUT model. J Hum Earth Future 5:85–99. https://doi.org/10.28991/HEF-2024-05-01-07

Abendroth A, Bender B (2024) Ein Werkzeug zur analyse von Komplexität von low-code und no-code Add-ons: Ein design science ansatz. HMD Praxis Wirtschaftsinformatik 61(5/2024):1180–1212. https://doi.org/10.1365/s40702-024-01109-9

Abendroth A, Oparin L, Pottek T (2026) Marktüberblick: interne ERP-low-code-plattformen. ERP Manag 22(2026):32–43

Abendroth A, Bender B (2025) Bridging the Gap: Low-Code Platforms and the Future of ERP Customization. AMCIS 2025 Proceedings, 28. https://aisel.aisnet.org/amcis2025/intelfuture/intelfuture/28/

Adrian B, Hinrichsen S, Nikolenko A (2020) App Development via Low-Code Programming as Part of Modern Industrial Engineering Education. In: Nunes IL (ed) Advances in Human Factors and Systems Interaction. Springer International Publishing, pp 45–51

Akundi A, Mondragon O (2022) Model based systems engineering—a text mining based structured comprehensive overview. Syst Eng 25(1):51–67. https://doi.org/10.1002/sys.21601

Alfonso I, Conrardy A, Cabot J (2025) Towards the Interoperability of Low-Code Platforms. In: Pufahl L, Rosenthal K, España S, Nurcan S (eds) Intelligent Information Systems, vol 557. Springer Nature Switzerland, pp 3–11

Avishahar-Zeira A, Lorenz DH (2023) Could No-Code Be Code? Toward a No-Code Programming Language for Citizen Developers. In: Proceedings of the 2023 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Onward! 2023, 103–119. https://doi.org/10.1145/3622758.3622893

Awasi M (2025) AI Agents for Automated Design and Compliance Verification in Buildings and Infrastructure: A Case Study Approach with No-Code/Low-Code Implementation. In: Proceedings of Digital Frontiers in Buildings and Infrastructure International Conference Series, 1–12.

Barricelli BR, Cassano F, Fogli D, Piccinno A (2019) End-user development, end-user programming and end-user software engineering: a systematic mapping study. J Syst Softw 149:101–137

Baumgarten C, Elshan E, Endl R (2025) Low Code und Citizen Development für eine erfolgreiche Digitalisierung. Springer Nature Link. https://doi.org/10.1007/978-3-658-48401-9

Bender B, Bretschneider S, Fattah-Weil J (2024) Advances in Demand Forecasting: A Systematic Review of Methods, The Role of AI, and Data Strategies in Manufacturing. AMCIS 2024 Proceedings, 7. https://aisel.aisnet.org/amcis2024/stratcompis/stratcompis/7

Bernsteiner R, Schlögl S, Ploder C, Dilger T, Brecher F (2022) Citizen vs. Professional Developers: Differences and Similarities of Skills and Training Requirements for Low Code Development Platforms. In: ICERI2022 Proceedings, 4257–4264. https://doi.org/10.21125/iceri.2022.1036

Beynon-Davies P, Williams MD (2003) The diffusion of information systems development methods. J Strateg Inf Syst 12(1):29–46. https://doi.org/10.1016/S0963-8687(02)00033-1

Bock AC, Frank U (2021) Low-Code platform. Bus Inf Syst Eng 63(6):6. https://doi.org/10.1007/s12599-021-00726-8

Borchers J (2018) Agile und DevOps treffen auf Regulatorik—Konfrontation oder Kooperation. GI-Fachausschuß Management Der Anwendungsentwicklung Und-Wartung, 42. Rundbrief, Jahrgang 2024(1), 12.

Bragança A, Azevedo I, Bettencourt N, Morais C, Teixeira D, Caetano D (2021) Towards supporting SPL engineering in low-code platforms using a DSL approach. In: Proceedings of the 20th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, 16–28. https://doi.org/10.1145/3486609.3487196

Bragg SD, Driskill CG (1994) Diagrammatic-graphical programming languages and DoD-STD-2167A. In: Proceedings of AUTOTESTCON '94, 211–220. https://doi.org/10.1109/AUTEST.1994.381508

vom Brocke J, Alexander S, Niehaves B, Niehaves B, Reimer K, Plattfaut R, Cleven A (2009) Reconstructing the giant: On the importance of rigour in documenting the literature search process. In: ECIS 2009 Proceedings, 14.

Bucaioni A, Cicchetti A, Ciccozzi F (2022) Modelling in low-code development: a multi-vocal systematic review. Softw Syst Model 21(5):5. https://doi.org/10.1007/s10270-021-00964-0

Burnett M (2009) What Is End-User Software Engineering and Why Does It Matter? In: Pipek V, Rosson MB, De Ruyter B, Wulf V (eds) End-User Development, vol 5435. Springer Berlin Heidelberg, pp 15–28

Cabot J (2020) Positioning of the low-code movement within the field of model-driven engineering. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 1–3. https://doi.org/10.1145/3417990.3420210

Carroll N, Ajimati M (2024) Unleashing the Power of Hyperagility in Citizen Development. Citizen Develpment Lab: Technical Report, University of Galway

Colantoni A, Berardinelli L, Wimmer M (2020) DevOpsML: Towards modeling DevOps processes and platforms. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 1–10. https://doi.org/10.1145/3417990.3420203

Cooper HM (1988) Organizing knowledge syntheses: a taxonomy of literature reviews. Knowl Soc 1(1):104–126. https://doi.org/10.1007/BF03177550

Coutaz J (2010) User interface plasticity: Model driven engineering to the limit! In: Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 1–8. https://doi.org/10.1145/1822018.1822019

Craven PV (2016) What Is a Computer Language? In: Craven PV (ed) Program Arcade Games: With Python and Pygame. Apress, pp 33–40

Curty S, Härer F, Fill HG (2022) Blockchain Application Development Using Model-Driven Engineering and Low-Code Platforms A Survey. In: Augusto A, Gill A, Bork D, Nurcan S, Reinhartz-Berger I, Schmidt R (eds) Enterprise, Business-Process and Information Systems Modeling. Springer International Publishing, pp 205–220

Curty S, Harer F, Fill H (2023) Design of blockchain-based applications using model-driven engineering and low-code/no-code platforms: a structured literature review. Softw Syst Model. . https://doi.org/10.1007/s10270-023-01109-1

da Cruz MAA, de Paula HTL, Caputo BPG, Mafra SB, Lorenz P, Rodrigues JJPC (2021) OLP—a RESTful open low-code platform. Future Internet 13(10):10. https://doi.org/10.3390/fi13100249

DeSilva DI, Ranathunga RAAL, Shangavie R (2023) Quality Assurance in Low-Code/No-Code Development. In: 2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), 1–7. https://doi.org/10.1109/ICSES60034.2023.10465268

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? Softw Syst Model 21(2):437–446. https://doi.org/10.1007/s10270-021-00970-2

Dobler M, Meierhofer J, Frick K, Bentele M (2022) On the Use of Low-Code and No-Code Tools for Teaching Data Science in Applied Industrial and University Settings. In: 2022 IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association For Management of Technology (IAMOT) Joint Conference, 1–8. https://doi.org/10.1109/ICE/ITMC-IAMOT55089.2022.10033266

Elshan E, Binzer B, Winkler TJ (2025) From software users to software creators: an exploration of the core characteristics of the citizen developer role and the related re- and upskilling programs. Bus Inf Syst Eng 67(1):31–53. https://doi.org/10.1007/s12599-024-00915-1

Fattah-Weil J, Abendroth A, Stoffels J, Kaan P, Horn HF, Karim A, Gronau N (2025) Whitepaper: Der ERP-Trendradar – Zukünftige Technologien und strategische Handlungsempfehlungen für ERP-Systeme [ERP-Trendradar]. Lehrstuhl für Wirtschaftsinformatik, insb. Prozesse und Systeme.

Favre JM, De Oliveira RA, Sottet JS, Quast M (2023) In Search of the Essence of No-Code – Elements of Data Modeling. In: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 878–887. https://doi.org/10.1109/MODELS-C59198.2023.00138

Fuggetta A (1993) A classification of CASE technology. Computer 26(12):25–38. https://doi.org/10.1109/2.247645

Gadde A (2025) Democratizing software engineering through generative AI and vibe coding: the evolution of no-code development. J Comput Sci Technol Stud 7(4):556–572. https://doi.org/10.32996/jcsts.2025.7.4.66

Gartner (2022). Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023. Gartner. https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023

Gawer A, Cusumano MA (2014) Industry platforms and ecosystem innovation. J Prod Innov Manag 31(3):417–433. https://doi.org/10.1111/jpim.12105

Girard J, Girard J (2015) Defining knowledge management: toward an applied compendium. Online J Appl Knowl Manag 3:1–20

Explosion AI GmbH Honnibal M, Montani I (2025) spaCy · Industrial-strength Natural Language Processing in Python. https://spacy.io/

Goasduff L (2021). Gartner Says Cloud Will Be the Centerpiece of New Digital Experiences. Gartner. https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences

Gronau N (2022) Geschäftsprozessmanagement in Wirtschaft und Verwaltung, Analyse, Modellierung und Konzeption (3.). GITO mbH Verlag.

Haddaway NR, Grainger Matthew J, Gray Charles T (2021) citationchaser: An R package for forward and backward citations chasing in academic searching (Version 0.0.3) [Computer software]. https://doi.org/10.5281/zenodo.4543513

Hailpern B, Tarr P (2006) Model-driven development: the good, the bad, and the ugly. IBM Syst J 45(3):451–461. https://doi.org/10.1147/sj.453.0451
Haleem A, Javaid M, Singh RP, Rab S, Suman R (2021) Hyperautomation for the enhancement of automation in industries. Sensors Int 2:100124. https://doi.org/10.1016/j.sintl.2021.100124
Haskins C (2011) A historical perspective of MBSE with a view to the future. INCOSE Int Symp 21(1):493–509. https://doi.org/10.1002/j.2334-5837.2011.tb01220.x
Henkel M, Stirna J (2010) Pondering on the Key Functionality of Model Driven Development Tools: The Case of Mendix. In: Forbrig P, Günther H (eds) Perspectives in Business Informatics Research. Springer, pp 146–160
Hirzel M (2023) Low-code programming models. Commun ACM 66(10):76–85. https://doi.org/10.1145/3587691
Hughes D (2002) Fundamentals of Computer Science Using Java. Jones & Bartlett Learning
Hurlburt G (2021) Low-code, no-code, what's under the hood? IT Prof 23(6):6. https://doi.org/10.1109/MITP.2021.3123415
Hyun CY (2019) Design and implementation of a low-code/no-code system. Int J Adv Smart Converg 8(4):188–193. https://doi.org/10.7236/IJASC.2019.8.4.188
IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12–1990, 1–84. IEEE Std 610.12–1990. https://doi.org/10.1109/IEEESTD.1990.101064
Jiang W, Hu F (2025) Artificial intelligence agent-enabled predictive maintenance: conceptual proposal and basic framework. Computers 14(8):329. https://doi.org/10.3390/computers14080329
Jost B, Ketterl M, Budde R, Leimbach T (2014) Graphical programming environments for educational robots: Open roberta-yet another one? 2014 IEEE International Symposium on Multimedia, 381–386. https://ieeexplore.ieee.org/abstract/document/7033055/
Kahanwal B (2013) abstraction level taxonomy of programming language frameworks. Int J Program Lang Appl 3(4):1–4
Käss S, Strahringer S, Westner M (2022) Drivers and Inhibitors of Low Code Development Platform Adoption. In: 2022 IEEE 24th Conference on Business Informatics (CBI), 196–205. https://doi.org/10.1109/CBI54897.2022.00028
Ko AJ, Abraham R, Beckwith L, Blackwell A, Burnett M, Erwig M, Scaffidi C, Lawrance J, Lieberman H, Myers B, Rosson MB, Rothermel G, Shaw M, Wiedenbeck S (2011) The state of the art in end-user software engineering. ACM Comput Surv 43(3):1–44. https://doi.org/10.1145/1922649.1922658
Kokala A (2022) Low-code platforms in BPM: how workflow and rules engines enable citizen development. Int J Sci Res Arch 6:343–346. https://doi.org/10.30574/ijsra.2022.6.1.0145
Krouwel MR, Op't Land M, Proper HA (2022) Generating Low-Code Applications from Enterprise Ontology. In: Barn BS, Sandkuhl K (eds) The Practice of Enterprise Modeling. Springer International Publishing, pp 18–32
Lebens M (2025) From Novice to Creator: Examining Student Perspectives on Low-Code Development Environments. In: Arai K (ed) Advances in Information and Communication. Springer Nature Switzerland, pp 332–351
Lethbridge TC (2021) Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering. In: Margaria T, Steffen B (eds) Leveraging Applications of Formal Methods, Verification and Validation. Springer International Publishing, pp 202–212
Li L, Wu Z (2022) How Can No/Low Code Platforms Help End-Users Develop ML Applications? - A Systematic Review. In: Chen JYC, Fragomeni G, Degen H, Ntoa S (eds) HCI International 2022 – Late Breaking Papers: Interacting with eXtended Reality and Artificial Intelligence. Springer Nature Switzerland, pp 338–356
Lieberman H, Paternò F, Klann M, Wulf V (2006) End-User Development: An Emerging Paradigm. In: Lieberman H, Paternò F, Wulf V (eds) End User Development. Springer, Netherlands, pp 1–8
Liu Y, Chen J, Bi T, Grundy J, Wang Y, Yu J, Chen T, Tang Y, Zheng Z (2025) An Empirical Study on Low-Code Programming using Traditional vs Large Language Model Support. *arXiv.Org*. Publicly Available Content Database (3303157471). https://www.proquest.com/working-papers/empirical-study-on-low-code-programming-using/docview/3303157471/se-2?accountid=14491
Lundell B, Lings B (2004) Changing perceptions of CASE technology. J Syst Softw 72(2):271–280. https://doi.org/10.1016/S0164-1212(03)00087-6
Martin J (1991) Rapid application development. Macmillan Publishing Co. Inc
Martinez, E., & Cisterna, D. (2023). Using Low-Code and Artificial Intelligence to Support Continuous Improvement in the Construction Industry. In: Proceedings of the 31st Annual Conference of the International Group for Lean Construction (IGLC31), 197–207. https://doi.org/10.24928/2023/0236

Martinez E, Pfister L (2023) Benefits and limitations of using low-code development to support digitalization in the construction industry. Autom Constr 152:104909. https://doi.org/10.1016/j.autcon.2023.104909

Martínez-Lasaca F, Díez P, Guerra E, De Lara J (2025) A model and workflow-driven approach for engineering domain-specific low-code platforms and applications. Softw Syst Model. https://doi.org/10.1007/s10270-025-01308-y

Matook S, Maggie Wang Y, Koeppel N, Guerin S (2023) Metacognitive skills in low-code app development: work-integrated learning in information systems development. J Inf Technol. https://doi.org/10.1177/02683962231170238

Matvitskyy O, Akash J, Davis K, Leow A (2025) Magic Quadrant for Enterprise Low-Code Application Platforms [Magic Quadrant]. Gartner. (G00824523).

Metrôlho J, Araújo R, Ribeiro F, Castela N (2019) An approach using a low-code platform for retraining professionals to ICT. In: 11th International Conference on Education and New Learning Technologies, 7200–7207.

Nagy C, Vidács L, Ferenc R, Gyimóthy T, Kocsis F, Kovács I (2011) Complexity Measures in 4GL Environment. In: Murgante B, Gervasi O, Iglesias A, Taniar D, Apduhan BO (eds) Computational Science and Its Applications—ICCSA 2011. Springer, pp 293–309

Noack K, Fedtke S (2025). Paradigmenwechsel Low-Code: Softwareentwicklung (fast) ohne Programmierung (1.st). Springer Nature Link.

Orlikowski WJ (1993) CASE tools as organizational change: investigating incremental and radical changes in systems development. MIS Q 17(3):309–340. https://doi.org/10.2307/249774

Picek R (2023) Low-code/No-code Platforms and Modern ERP Systems. 2023 International Conference on Information Management (ICIM), 44–49. https://doi.org/10.1109/ICIM58774.2023.00014

Pinho D, Aguiar A, Amaral V (2023) What about the usability in low-code platforms? A systematic literature review. J Comput Lang 74:101185. https://doi.org/10.1016/j.cola.2022.101185

Piridi S, Asundi S, Hyatt DJC (2025) Hyperautomation with Power Platform: merging AI, RPA, and low-code for business efficiency – exploring how AI Builder, Power Automate, and Dataverse can drive end-to-end enterprise automation. Int J Adv Eng Res Sci 12(4):54–69. https://doi.org/10.22161/ijaers.124.7

Pöppler N, Abendroth A (2025) Marktübersicht: Low-Code-Plattformen. ERP Management, 21(2025). https://erp-management.de/marktuebersichten/low-code-plattformen/

Ray PP (2025) A Review on Vibe Coding: Fundamentals, State-of-the-art, Challenges and Future Directions. TechRxiv. https://www.techrxiv.org/doi/full/https://doi.org/10.36227/techrxiv.174681482.27435614/v1

Richardson C, Rymer J (2014) New Development Platforms Emerge For Customer-Facing Application. Forrester. https://www.forrester.com/report/new-development-platforms-emerge-for-customer-facing-applications/RES113411

Ritterbusch GD, Teichmann MR (2023) Defining the metaverse: a systematic literature review. IEEE Access 11:12368–12377. https://doi.org/10.1109/ACCESS.2023.3241809

Rymer JR, Koplowitz R, Leaders SA, Mendix K, are Leaders S, ServiceNow G, Performers S, MatsSoft W, are Contenders T (2019). The forrester wave™: Low-code development platforms for ad&d professionals, q1 2019. Forrester Research.

Sahay A, Di Ruscio D, Pierantonio A (2020) Understanding the role of model transformation compositions in low-code development platforms. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 1–5.

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.

Sanchis R, García-Perales Ó, Fraile F, Poler R (2020) Low-code as enabler of digital transformation in manufacturing industry. Appl Sci 10(1):1

Sarabu AR (2025) iPaaS: revolutionizing enterprise integration in distributed commerce. Eur J Comput Sci Inf Technol. https://doi.org/10.37745/ejcsit.2013/vol13n134764

Simonsen J, Robertson T (2013) Routledge international handbook of participatory design, vol 711. Routledge, New York

Skog, K. (2005). From Binary Strings to Visual Programming: A Nordic perspective on history of programming and programming languages. In J. Bubenko, J. Impagliazzo, & A. Sølvberg (Eds.), *History of Nordic Computing* (Vol. 174, pp. 297–310). Springer US. https://doi.org/10.1007/0-387-24168-X_27

Stapleton J (1997). DSDM, dynamic systems development method: The method in practice. Cambridge University Press. https://books.google.com/books?hl=de&lr=&id=kp656t4p7soC&oi=fnd&pg=PR13&dq=Stapleton,+J.,+1997.+DSDM%E2%80%94Dynamic+Systems+Development+Method:+the+Method+in+Practice,+Addison-Wesley,+Harlow,+England.&ots=TAGeMuUTWe&sig=5wTydyACXIEztSbg5RZy7Mof6tI

Tarasiev A, Filippova M, Aksyonov K, & Aksyonova, O. (2018). Developing Prototype of CASE-Tool to Create Automation Systems Based on Web Applications Using Code Generation. 2018 Dynamics of Systems, Mechanisms and Machines (Dynamics), 1–4. https://doi.org/10.1109/Dynamics.2018.8601443

Tisi M, Mottu JM, Kolovos DS (2019) Lowcomote: Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms.

Totterdale RL (2018) Case study: the utilization of low-code development technology to support research data collection. Issues Inf Syst 19(2):2

Vailshery LS (2024). Topic: Low-code and no-code platforms. Statista. https://www.statista.com/topics/8461/low-code-and-no-code-platforms/

Vincent P, Iijima K, Driver M, Wong J, Natis Y (2019) Magic Quadrant for Enterprise Low-Code Application Platforms (pp. 1–35) [Magic Quadrant]. Gartner.

Viswanadhapalli V (2025) The future of intelligent automation: how low-code/no-code platforms are transforming AI decisioning. Int J Eng Comput Sci 14(1):26803–26825

Waszkowski R (2019) Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine 52(10):10. https://doi.org/10.1016/j.ifacol.2019.10.060

Woo M (2020) The rise of no/low code software development—no experience needed? Engineering 6(9):9. https://doi.org/10.1016/j.eng.2020.07.007

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.