The Low-Code Phenomenon: Mapping the Intellectual Structure of Research

Syed Asad Ali Naqvi
Leuphana University of Lüneburg
asadali.naqvi@outlook.com

Markus P. Zimmer
Leuphana University of Lüneburg
markus.zimmer@leuphana.de

Paul Drews
Leuphana University of Lüneburg
paul.drews@leuphana.de

Kristina Lemmer
Leuphana University of Lüneburg
kristina.lemmer@leuphana.de

Rahul C. Basole
Accenture Data & AI
rahul.basole@accenture.com

Abstract

The term low-code has been closely associated with simplifying and accelerating software development. Driven by the idea that low-code can help to meet the increased digitalization demands, the low-code phenomenon is rising in academia and industry. This resulted in an immense increase in publications on low-code, posing the question of what research streams characterize the low-code literature. We conducted a bibliometric analysis based on 725 articles. Out of these 725 articles, we selected 105 articles with the term "low-code" in the title or abstract. Our contribution is to clarify the conceptual understanding of low-code by identifying six research streams, namely, origins of low-code within software engineering, low-code as an enabler for emerging software engineering trends, workplace transformation, establishing low-code methodologies, understanding low-code adoption and leveraging low-code for digital transformation. We conclude with future research directions that still need to be explored within the low-code literature.

Keywords: Low-code, intellectual structure, bibliometrics, visualization, low-code platform (LCP), LCP affordances.

1. Introduction

The low-code phenomenon refers to an emerging trend in software development to help organizations overcome digital transformation challenges and to meet their increasing digitalization needs. Low-code is perceived as a way to democratize software development, making it more accessible to non-IT professionals. Low-code is intended to be intuitive and simple and can be used without significant training [1]. Such characteristics enable non-IT professionals to carry out tasks that software developers would have typically handled. These non-IT professionals are often called citizen developers, and their enablement via low-code is known as citizen development [2]. Despite the widespread popularity of low-code platforms (LCP), the term low-code is still ambiguous [3, 4]. This is mainly because research on low-code originates from multiple disciplines and various emerging software engineering trends [5, 6, 7, 8]. A shared understanding of what constitutes low-code will help researchers to advance their theoretical understanding and aid practitioners in communicating its value.

The literature on the nascent low-code field has rapidly grown during the past decade. Regardless, low-code does not have a standard definition, which can hinder our understanding and adoption of low-code [5, 8]. Given low-code's various origins, we argue that we need to take a broad perspective to understand the low-code phenomenon. Indeed, while the number of studies on low-code surges, we lack an intellectual structure of this work. An intellectual structure maps existing literature on a phenomenon and aggregates it to research streams. Such a structure can direct research and by this, instruct and influence future scholarship. This makes it a vital tool for developing an understanding within a certain field [9]. We thus conduct a literature review of existing work on the low-code phenomenon to build an intellectual structure of the low-code literature. We aim to tackle the ambiguity of the low-code phenomenon by mapping our understanding to its origins. This produces useful insights into the emergence and evolution of low-code by evaluating links between ideas and concepts across existing work. The intellectual structure also highlight different disciplines' perspectives on low-code.

We investigate the following research question: *What research streams characterize the low-code literature*? Our work makes three contributions. First, we uncover the theoretical origins of the low-code phenomenon by creating a curated collection and finding publications from multiple research disciplines. Second, we identify patterns within the existing low-code literature and map the intellectual structure of low-code research by employing an automated literature analysis. Third, we identify six low-code research streams by analyzing 105 low-code publications.

2. Theoretical Foundations and Related Work

Low-code aims to reduce the use of computer programming by offering ways to develop software intuitively. Moreover, it broadens the scope by accepting that some programming would remain necessary for software development; thus, LCPs assist both citizen and software developers in software development [8]. LCP enables software development using the low-code approach. In this paper, we refer to "low-code" as the primary concept and "LCPs" when referring to low-code platforms. Low-code is often interchangeably used with the term no-code. The concept of no-code states that software applications can be developed without any programming through a design tool. In previous academic literature, this concept resembled end-user programming, that focuses on how end-users, mainly citizen developers, can develop an application [10].

Software engineering research associates low-code with the concept of Model-driven engineering (MDE) which is a software development approach that relies on models as key artefacts. MDE divides applications into manageable modules that can be tested individually and seamlessly merged, resulting in fewer mistakes and higher quality [11]. When the low-code phenomenon emerged as a research topic, researchers viewed it as a new form of end-user programming and an approach that supports MDE. Hence, the initial research discourse on low-code linked it to existing concepts, including MDE, end-user programming and rapid application development. However, since the publications on low-code increased tremendously, many studies have investigated low-code capabilities [12] and the socio-technical factors affecting the adoption of low-code [4].

We found three existing literature reviews on low-code. The first literature review on low-code was conducted by [13], who reported that most studies on low-code offer a technological view and very limited (only three studies at that time) studies of the social aspects associated with low-code. Therefore, there is still a need for a comprehensive view of how low-code is viewed by business users, such as in the context of citizen development. Another literature review on the usability of low-code platforms was conducted by [14]. They identified common characteristics between low-code and no-code platforms and twelve usability elements related to low-code and no-code reported in the existing literature, mainly from a technical view. To explain the drivers and barriers to low-code adoption, [15] conducted a literature review. They identified thirteen factors that hinder and seven aspects that support low-code adoption. They used an organizational context to identify these factors and lacked focus on individual user decisions to use LCPs. However, none of these literature reviews provides a comprehensive overview of existing research streams on low-code.

3. Research Methodology

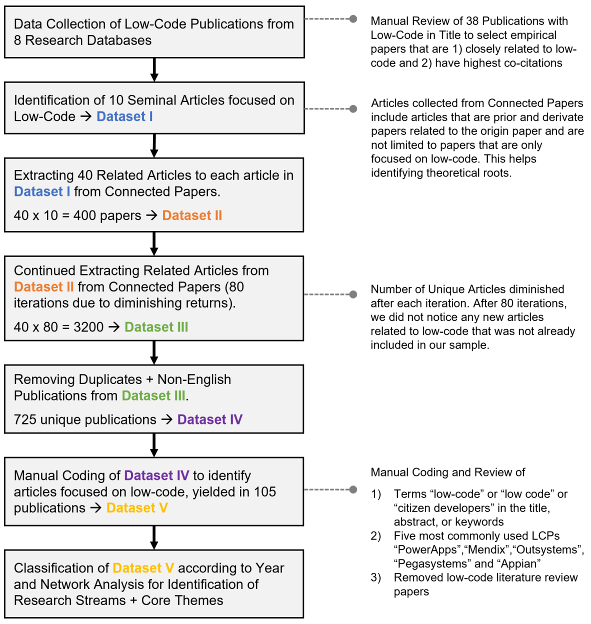
To understand the intellectual structure of the low-code phenomenon, a fundamental first step is to identify salient articles. We started our search by identifying seminal papers with "low-code" or "low code" in their title using eight databases, including ACM Digital Library, AISeL, IEEE Xplore Digital Library, JSTOR, Science Direct, Emerald, SpringerLink, and Wiley Online Library. We choose these databases, as these offer an extensive range of peer reviewed publications from various disciplines. We focused the selection of publications on studies that explicitly deal with low-code development from a socio-technical perspective. Moreover, we included the existing three literature studies in the initial dataset, as they help to identify the relevant literature on low-code. Our initial dataset included ten papers [A1, A2, A3, A4, A5, A6, A7, A8, A9, A10]. Next, we leveraged *Connected Papers*[1], a web-based tool to help researchers find relevant academic papers to identify key prior and derivative works for a given focal paper. Connected Papers is linked with the semantic scholar paper corpus[2], enabling it to access millions of articles from various scientific subjects. Connected Papers outputs a visual graph and corresponding tabular data of relevant associated papers (prior and derivative)[3] weighted by a similarity metric, using co-citation and bibliographic coupling analysis. The graph lays out articles using a force-directed algorithm, placing similar articles closer to the focal paper and less similar articles further apart.

[1]www.connectedpapers.com

[2]www.semanticscholar.org

[3]Prior articles are the most cited by the papers in the graph, while derivative articles cite many of the papers in the graph.

The similarity metric implies that two articles with significantly overlapping citations and references are more likely to cover a similar topic area. Each graph for a given paper consists of 40 related publications, thus leading to an intermediate set of 400 publications. Using a subsequent Snowball technique [16], we continued to collect additional articles for each of these 400 publications as the focal article. As we continued the process, we observed that the number of unique articles was diminishing after each search. After 80 search iterations and removing duplicate publications, our final dataset yielded 725 publications. We summarize our data collection and analysis steps in Figure 1.


Data Collection of Low-Code Publications from 8 Research Databases

Manual Review of 38 Publications with Low-Code in Title to select empirical papers that are 1) closely related to low-code and 2) have highest co-citations

Identification of 10 Seminal Articles focused on Low-Code → Dataset I

Articles collected from Connected Papers include articles that are prior and derivate papers related to the origin paper and are not limited to papers that are only focused on low-code. This helps identifying theoretical roots.

Extracting 40 Related Articles to each article in Dataset I from Connected Papers.

40 x 10 = 400 papers → Dataset II

Continued Extracting Related Articles from Dataset II from Connected Papers (80 iterations due to diminishing returns).

40 x 80 = 3200 → Dataset III

Number of Unique Articles diminished after each iteration. After 80 iterations, we did not notice any new articles related to low-code that was not already included in our sample.

Removing Duplicates + Non-English Publications from Dataset III.

725 unique publications → Dataset IV

Manual Coding of Dataset IV to identify articles focused on low-code, yielded in 105 publications → Dataset V

Manual Coding and Review of

1) Terms "low-code" or "low code" or "citizen developers" in the title, abstract, or keywords
2) Five most commonly used LCPs "PowerApps","Mendix","Outsystems", "Pegasystems" and "Appian"
3) Removed low-code literature review papers

Classification of Dataset V according to Year and Network Analysis for Identification of Research Streams + Core Themes

Figure 1. Data Collection and Analysis.

3.1. Network Construction

In order to understand the overall structure of low-code research, we created an integrative network dataset of all the individual publication similarity networks we extracted above. Nodes represent publications; edges represent the weighted similarity between any pair of papers.

3.2. Visualization

We used Gephi 0.10.1 [17], an open-source graph visualization software, to visualize the low-code development research network. We used OpenORD, a cluster emphasizing layout algorithm, to depict the overall structure [18]. We sized nodes proportionally to their betweenness centrality to distinguish prominent nodes. Edge thickness was encoded using the weighted similarity between two papers. We then applied Louvain's community detection algorithm to identify and color sub-communities in the ecosystem [19]. This allows us to differentiate among nodes by how closely related they are to each other.

4. Results

In the following, section 4.1 describes the evolution of low-code literature over time. Next, we explain the results of the network analysis in section 4.2, followed by section 4.3, which explains the research streams' classification and characteristics.

4.1. Classification by Year

After constructing the network in Gephi, we selected papers based on the inclusion criteria (see section 3) for further analysis. We searched for the terms "low-code" or "low code" or "citizen developers" in the title, abstract, or keywords within the 725 publications (nodes) manually. We also included five most commonly used LCPs ("PowerApps", "Mendix", "Outsystems", "Pegasystems" and "Appian"). Our search yielded a total of 105 publications.[4] The annual distribution of articles is shown in Figure 2. It is apparent from the results that there was a significant increase in articles from 2020. Only 12 articles have been published between 2014 and 2019, whereas 93 contributions have been published between 2020 and 2022, which accounts for approximately 89 percent of all studies in our sample of the low-code literature.

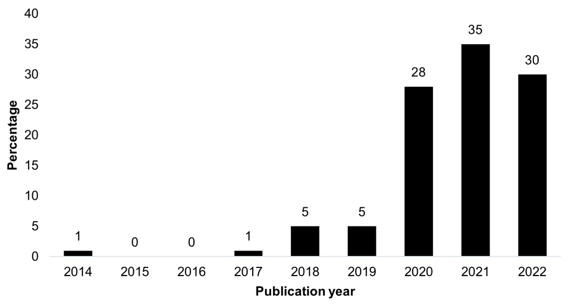


Figure 2. Number of Publications on Low-Code by Year (Dataset V).

[4]Given page length constraints, we offer a comprehensive reference list of articles included in our study in an online appendix, available at https://zenodo.org/record/8354060.

4.2. Classification by Network Analysis

Figure 3 visualizes the intellectual structure of the low-code literature.[5] The modularity analysis reveals nine distinct clusters derived from the co-citation and bibliographic coupling analysis. Each cluster, coloured differently, represents distinct research disciplines. Additionally, these clusters sometimes intertwine and/or overlap; some share similar assumptions but focus on various disciplines, implying that the research streams complement each other.

The central core of the network stems from the topics of LCPs affordances [A11, A12, A13, A14] and investigates the characteristics and challenges to low-code adoption [A15, A16, A17]. Most of these articles are practice-oriented, with micro analyses of specific LCPs characteristics or demonstrate LCP use cases. We observe multiple small clusters in the network's periphery that demonstrate low-code's wide range of applications. These include mostly technical papers investigating the role of low-code as an enabler for emerging software engineering trends for example, construction of recommender systems [A18, A19], and social media monitoring [6], improving technical capabilities of low-code [A20, A21] and enabling new software development practices including citizen development [A22, A23]. Some small clusters with similar theoretical origins were combined to understand better and analyze the underlying concepts and principles that govern low-code. The authors then conducted multiple rounds to interpret the constructed network. Next, analyzing the core themes (see table 1), using triangulation between the researchers, we formulated the names of the research streams. We conducted three iterations, until consensus was developed between all authors about the name of each research stream. For instance, research stream III has seminal works focusing on citizen development, which was proposed as the research stream name in the first round. However, after analyzing and discussing the second sub-group of research stream III, the authors identified the literature oriented towards new ways of simplifying jobs for both software developers and non-IT professionals. Hence, the name "workplace transformation" was concluded. This resulted in six research streams (see Figure 3) with a more cohesive and unified view of low-code literature, leading to a better understanding of low-code literature.

[5]A high resolution visualization can be downloaded from https://zenodo.org/record/8354060.

4.3. Classification by Research Streams within the Low-Code Literature

In the following, we highlight each research stream and the publications in these streams.

4.3.1. Research Stream I: Origins of Low-Code within Software Engineering.

The first research stream stems from the software engineering discipline. The central themes within this research stream are technical papers that analyze the similarities and differences of low-code with existing software engineering approaches and between LCPs. The level of analysis is mainly meso, i.e. either comparison of multiple LCPs (more than five) or evaluating LCPs characteristics on an abstract level. These include comparison of low-code with MDE [A19, A24, A37] and LCP specific use cases that compare the technical characteristics of LCPs [A29, A30].

A growing body of literature has investigated the parallels between the LCPs and MDE and reported that LCP foundations stems from the MDE. [A24]. According to [5], traditional MDE primarily rely on offline downloaded resources, resulting in scalability and integration challenges. LCP can support to resolve such challenges in MDE, as it is cloud based. LCP simplifies the changes in workflow and analysis, testing and deployment of models [5]. Whereas, MDE also leveraged cloud computing, GUI, and declarative programming, LCP adds an additional flexibility component by offering programming interfaces for accommodating complex requirements, instead of striving for complete elimination of programming [A26].

Other works within this research stream investigate how LCPs advances software engineering by integrating MDE, Cloud Computing, and Machine Learning [A37, A39, A40]. Within the IoT discipline, [A37] analyzed sixteen LCPs with MDE approaches and created a taxonomy of low-code features. They reported that compared to MDE approaches, LCP offer limited support for multi-view modeling, testing and analysis (ibit). [A29] discusses how LCPs can be leveraged for managing collaborative manufacturing and logistics environments. Remaining works within this research stream focus on LCP architecture [A8] and low-code testing [A35].

4.3.2. Research Stream II: Low-code as an Enabler for Emerging Software Engineering Trends.

The second research stream is a combination of analytical and technical papers, that develops an understanding on the role of low-code phenomenon in supporting

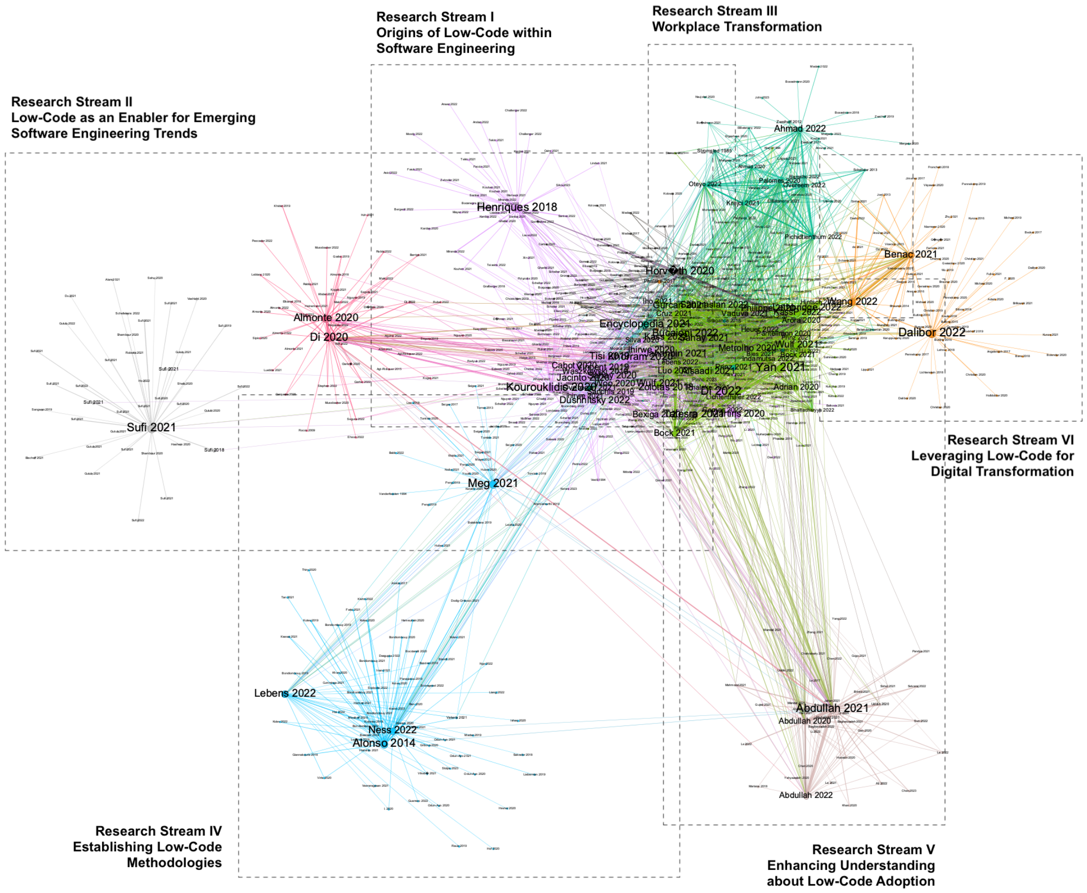


Figure 3. Intellectual Structure of Low-Code Research 2014-2022.

advances in software engineering [A42, A43, A44, A45], tackling socio-economic challenges and creating new employability trends [A50, A51, A52, A53]. This research stream is created by merging three peripheral clusters. The level of analysis is micro-level, i.e. most articles in this research stream study low-code either with focus on an explicit technology or a socio-technical challenge. Exploring the ways to improve the documentation for application landscapes, [A48] demonstrated the implementation of low-code automation for interactive visualizations and application documentation. Other work includes leveraging low-code for the integration of IoT devices and platforms [A46] and for test automation including Unit, API, System/End-to-End testing levels [A47], managing models for optimized performance [A43, A45].

Within the second sub-theme of this research stream, we identified three examples demonstrating how low-code helps in tackling socio-economic challenges. First, since during a recent pandemic COVID-19, the demand for digital applications is increased tremendously, [A53] illustrates the use of low-code for monitoring and maintaining health protocols. Second, [A52] highlights the usefulness of low-code for software sustainability, especially in public sector, since public sector strongly rely on customized software from consulting firms through public tenders [A52]. Finally, LCP can be leveraged for retraining individuals with STEM backgrounds and with basic IT knowledge to meet the increased digital

Table 1. Intellectual Structure of Low-Code Research 2014-2022.

#	Research Streams	Core Themes	Key Publications
I	**Origins of low-code within Software Engineering**	MDE	[A1, A24, A25, A26]
		LCP specific use cases	[A27, A28, A29, A30, A31, A32, A33, A34]
		Application Testing	[A35, A36]
		Interconnected digital technologies	[A37, A38, A39, A40]
II	**Low-code as an enabler for emerging software engineering trends**	Advances in Software Engineering	[A7, A41, A42, A43, A44, A45, A46, A47, A48, A49]
		Tackling socio-economic challenges	[A50, A51, A52, A53]
		Recommender systems	[A18, A36, A54]
		AI-based analysis and social media monitoring	[A2]
III	**Workplace transformation**	Citizen development	[A22, A23, A55, A56, A57, A58]
		Supporting domain-specific approaches, micro-services and APIs	[A59, A60, A61, A62, A63, A64, A65, A66, A67]
IV	**Establishing low-code methodologies**	Low-code development methodologies	[A20, A68, A69, A70, A71, A72, A73],
		Polyglot data access layer	[A21, A74]
V	**Enhancing understanding about low-Code adoption**	Characteristics and challenges of low-code adoption	[A4, A6, A9, A10, A15, A17, A75, A76, A77, A78, A79, A80, A81, A82, A83]
		LCP Affordances	[A3, A11, A12, A13, A14, A84, A85, A86, A87, A88, A89, A90, A91, A92, A93, A94]
		Barriers to low-code adoption	[A5, A95, A96, A97, A98, A99]
VI	**Leveraging low-code for Digital Transformation**	Enterprise application development	[A75, A100, A101, A102, A103]
		Digital Twins	[A104, A105]

transformation needs [A50, A51]. LCPs can benefit from recommender systems as well since such systems guide users with useful, customized recommendations based on the learnings from developing prior software applications [A18]. LCPs should also provide testing, debugging capabilities, maintainability, and backward compatibility [A85].

4.3.3. Research Stream III: Workplace Transformation.

The third research stream uncovers the role of low-code phenomenon in workplace transformation. The central works within this research stream are empirical and technical papers, with meso-level analysis for investigating how low-code enables new software development practices for workplace transformation. We identified that the low-code phenomenon is supporting the transformation in two ways. First, low-code enables citizen development, and collaboration between software developers and citizen developers. With its easy to use and intuitive features, LCPs empower citizen developers to shape their own environment [A22, A23, A56]. Moreover, LCPs empower software developers, since using LCPs they can save time and efforts on simple tasks and can instead concentrate their attention on other issues that call for in-depth technical expertise [A55, A57, A58]. Moreover, LCPs support developers to improve their understanding of the business, since several technical activities including application infrastructure, scalability, extensibility, data integrity are managed through LCPs [A55, A56].

Second, organizations can leverage low-code to streamline processes and improve workflow by dividing them into smaller, more manageable micro-services. Utilizing domain-specific approaches, micro-services, and APIs, LCPs provide valuable integration and communication capabilities between different applications, enabling task automation [A60, A61, A62, A64].

4.3.4. Research Stream IV: Low-Code Methodologies. The fourth research stream stems from the network periphery. It focuses on the low-code methodologies and compares it with traditional software development [A20, A70, A73]. Most articles in this stream are descriptive and follow the case study methodology [A20, A69, A71].

Recent studies evaluating low-code in educational institutions' process improvement [A71], and being used as an enabler for agile software development [A20] prove its usefulness. At the same time, due to the wide applications and high number of LCPs, it is challenging for organizations to select the LCP that best matches their needs [A72]. Other works within this research stream investigate Polyglot[6] capabilities of LCP and suggest ways for how LCP can improve such capabilities. [A21, A74].

4.3.5. Research Stream V: Enhancing Understanding about Low-Code Adoption. The fifth research stream focuses on explaining low-code adoption. Most articles in this stream use empirical approaches to conduct macro-level analyses, to investigate the LCP affordances, and challenges to low-code adoption [A15, A16, A17, A76, A79, A81, A83].

According to [20], LCPs enable fast application development, are easier to learn and use, compared to programming languages and can reduce IT costs. Moreover, LCPs also provide ready-to-use implementation units that enable agile software development. On the other hand, vendor lock-in and no access to source code are challenges that may hinder its adoption. Despite the emergence of LCPs, the need for skilled programming will exist, for customized requirements [A17]. Code written LCPs can be complex and difficult to understand. Thus, LCP should support modern software engineering practices such as literate coding, self-documentation, and automated testing.

According to Crunchbase[7], more than 1100 vendors sate their products feature low-code characteristics. Several studies have compared the characteristics of widely adopted LCP including Appian, Mendix, OutSystems and Microsoft Power Apps [A3, A11, A14, A84, A86, A87]. Comparing eight LCPs, Web portals, business process automation systems, and quality management, [A28] identified six characteristics of LCP as mandatory. These include graphical user interface, interoperability support with external services, application security, business logic specification mechanisms, Application build mechanisms and deployment support and two optional features including re-usability and scalability support. This demonstrates that LCPs functionalities may vary a lot and most of the LCPs may not offer optional features.

Some articles in this research stream report that LCPs may fall short to fulfill the promises and illustrate challenges related to LCP adoption. Studies show that applications developed on LCPs can be difficult to maintain and customize beyond the standard features offered by the LCPs [A97]. This is mainly because access to source code is often not provided by LCPs. Even in cases, where LCPs provide access to the code, modifying the automated generated code can result in errors upon the execution of application, leading to high efforts in troubleshooting [A99]. Whereas, LCPs promise to empower employees, in particular citizen developers, [A56] reported that application deployment phase in LCPs require in-depth technical knowledge and to be managed solely by software developers. Such situations may result in maintaining the IT-business gap. Moreover, even software developers may experience isolation, as they are dependent on the LCP vendor for application maintenance and troubleshooting [A56].

Since LCPs lack standardization, [20] reported that LCPs may have a steep learning curve. The simple and easy to use design of LCPs may also pose difficulty to software developers, who are interested in managing the application beyond prototyping. Moreover, LCP providers have strong focus on building enterprise applications, instead of establishing long-term ecosystem that facilitates application development. Other risks include lack of transparency and limitations to scalability and vendor-lock in. Additionally, implementation of advanced tools and capabilities including application monitoring, analytics, user engagement is often left open ended to customers, who may setup and configure several protocols themselves [A55].

4.3.6. Research Stream VI: Leveraging Low-Code for digital transformation. The sixth research stream evaluates the role of low-code for enabling digital transformation. It mainly comprises of technical papers and analyses at the meso-level. This stream is connected to other peripheral streams that focus on software engineering via central research stream V.

The low-code phenomenon is being hailed as the

[6]Polyglot is the ability to communicate through multiple programming languages

[7]www.crunchbase.com

key infrastructure for digital transformation since it enables application development at a faster pace. The traditional software development struggle to keep-up with the increasing demands for digital applications because of their long development cycles and complex deployment processes. LCPs offer pre-built components and templates that optimize productivity and streamline the development process [A101, A102]. Moreover, LCPs offer a diverse range of applications, including general-purpose, process-based, database, request handling, and mobile-based applications. Due to such capabilities, LCPs enable flexibility and scalablity for enterprise application development. [A75].

Analyzing the efficiency of three LCPs namely, OutSystems, Microsoft, and Salesforce, [7] reported differences in the use cases, particularly related to the applications integration. Moreover, they reported the risks involved with low-code, such as the difficulty in choosing the best platform because of the high number of available LCPs. This stems another challenge, that LCPs lack a unified framework. To address this challenge [A101], explained a prototype LCP framework that offers one-click deployment and comprehensive data monitoring. Thus, LCP make it easier to develop digital applications for niche cases, for instance predictive maintenance application [A101].

5. Discussion

Based on the identified research streams, we develop the following definition for the low-code phenomenon to offer a bird's eye view of the low-code literature. *The low-code phenomenon* ***originates from software engineering and integrates software engineering tools*** *to offer unified capabilities for* ***simplifying and advancing software development, and workplace transformation,*** *thus, contributing to the* ***digital transformation****. At the same time, we need to establish and* ***enhance low-code methodologies to promote low-code adoption****.*

This definition can help to develop a common understanding of the low-code phenomenon. Future researchers can use this definition to orient their work within the low-code phenomenon. By analyzing the evolution of the low-code phenomenon, we identify that the initial works on low-code stem from software engineering disciplines (research streams I and II). Then it dispersed under other research disciplines. This can be because, in the beginning, inspired by the analyst reports [A34], researchers mainly studied the low-code phenomenon from technical perspectives [A48]. As researchers delved into the various aspects of low-code, MDE was identified as a technology which is closely related to low-code [A24]. Publications within the research stream we identified share many similar characteristics that low-code and MDE [5]. On the other hand, we do not identify publications comparing low-code phenomenon with EUP, that illustrates low-code phenomenon is not about moving away from code, rather it focuses on facilitating both software and citizen developers.

The practice-driven research in research stream V focused on comparing low-code platforms [A7]. Most studied platforms include Microsoft PowerApps, Mendix and OutSystems [12]. While analyzing the recent work on LCPs, we identified that several publications refer to the shortcomings of low-code. Such work aims to propose approaches to how low-code can overcome these limitations [A21, A60]. From the visualization, we observe that whereas low-code is mainly discussed from the software engineering context, few studies focus on the digital transformation contexts. Research streams II, IV, V, and VI are very distinct and least interconnected on the peripheries. Research streams I and III are strongly interconnected and overlapping. Moreover, we identified a dominance of descriptive papers and a need for more substantial studies grounded in appropriate theoretical lenses.

Our results from the central research stream V demonstrate that the low-code adoption phenomenon fulfil several adoption characteristics discussed in affordance theory [21, 22]. The design of LCPs is simple and intuitive, with cues guiding the users about the next steps and best practices. Studies have reported that MDE has shared similar characteristics, including automated code generation and built-in deployment with low-code. However, the distinct characteristic of low-code is it reduces the need for programming [5]. This implies that due to the easy-to-use and drag-and-drop characteristics of low-code, both software and citizen developers are encouraged to use it, resulting in a faster adoption [A89]. At the same time, if the features of low-code are perceived as difficult to use, this can act as a major barrier hindering its adoption [A22]. While many studies have reported difficulties such as scalability, and security, researchers are also investigating approaches to how low-code can overcome these challenges [A35, A98]. This illustrates that, despite the widespread popularity in practice, low-code adoption is yet an emerging phenomenon, with most research in low-code literature comprising practice papers with micro-level analyses focusing on specific LCP or use cases [A33, A69].

The detailed classification in six research streams from our work can help practice to determine how different research disciplines study the low-code phenomenon. While the discussed articles across these streams acknowledge low-code as the next big evolution in software engineering, we identified that emerging trends in practice, including generative AI, are currently not investigated in relation to low-code. Moreover, we observe that while there has been significant growth in studies examining the LCP characteristics, challenges, and affordances, empirical research on the dynamics and evolution of low-code ecosystems is still relatively nascent.

Several studies argued that LCP is not a new or novel technology [8], rather what is unique about the low-code phenomenon is it offers traditional software development elements, including IDEs, databases, GUI, code compilation and testing interfaces, through a single environment as a one-stop shop [A7]. Leveraging such capabilities, LCPs help advance the software engineering discipline in emerging areas, including the machine learning discipline, social media monitoring, and support for the rapid security requirements of organizations [A98, A38]. Reduced coding efforts, faster application development, and a manageable learning curve for software developers are prominent benefits of low-code reported in the literature [A30]. On the other hand, studies report that some software developers perceive low-code as a barrier to their creativity and have concerns that citizen development would challenge their identity [4]. However, we lack explanations of how low-code adoption would impact the identity of software developers. Future research is needed to understand the perceived challenges and risks associated with low-code adoption from macro perspectives, i.e. cross-industry perspectives, interchangeability between LCPs, and how the low-code phenomenon challenges the identity of software developers.

In past years, research on low-code has progressed substantially. Despite this widespread study of low-code, we lack an understanding of what research streams characterize the low-code phenomenon. Indeed, existing work provides for ambiguity in understanding the low-code phenomenon. We found that this ambiguity roots in the intellectual structure underpinning low-code research's evolution. We show this evolution by mapping this intellectual structure, and by this, we contribute to low-code research in three ways. First, we developed a curated dataset within which we identified articles from multiple research disciplines that constitute the theoretical roots of existing work on low-code. Second, we visualize this rooting in a network graph representing the results of bibliometrics and co-citation analysis. Third, we present six research streams we identified based on our bibliometrics and co-citation analysis to unpack how low-code literature is interconnected across disciplines. These contributions serve as a starting point for developing a research agenda for future low-code research at the fringes of the different research streams. For practice, we resolve the ambiguity associated with low-code by explaining its origins and offering insights into the applications of low-code in emerging software engineering trends and workplace transformation. Using the identified six research streams, practitioners can gain a comprehensive understanding of low-code and its capabilities. This can lead to more effective use of low-code, allowing practitioners to create robust applications and identify the respective use cases.

By leveraging a visualization approach for understanding the intellectual structure, we provide a better understanding of the connections with other disciplines and uncover the theoretical origins of the low-code phenomenon. We acknowledge that our work may have certain limitations, each of which present an opportunity for future research. Visualization for literature analysis is a fruitful approach because it allows for analyzing large amounts of data that are very difficult to explore using traditional systematic literature review approaches. Visualizations are useful in providing a broader frame of reference, highlighting the most important work within the literature, and uncovering the heterogeneity by identifying different sub-fields. At the same time, we acknowledge that a visualization-only approach can treat grey literature and short articles in the same way as peer-reviewed unless explicitly encoded. We mitigated this problem using a mixed-method approach by combining computer-aided literature analysis with manual review. Future research can leverage natural language processing and text analytic methods to analyze the network more deeply and develop more refined research stream clusters and their core topics.

Another potential limitation is the number and types of sources we included. We started our search process using eight leading databases. It may be possible that that some disciplines may not be comprehensively represented in these databases. To the best of our knowledge, we used keywords that best encompass the low-code phenomenon. However, the list of keywords could be extended to account for tangential concepts. From a visualization method approach, our goal was to unpack the intellectual structure of the low-code phenomenon. We used a visual layout that focuses on identifying core and peripheral structure as well as clusters within the the low-code literature. Future research can leverage different types of visual layouts and attributes to reveal other characteristics, for instance, evolution of clusters, disciplinary points of focus, or research lineages.

6. Conclusion

The low-code phenomenon has rapidly grown during the past years to help organizations tackle their digital transformation challenges. However, the lack of a standard definition and understanding of how low-code literature is interconnected to other disciplines can hinder the adoption. Our study aims at mapping the intellectual structure of the low-code phenomenon. We leveraged visualization for literature analysis to allow for a better understanding of the relationships and theoretical roots of the low-code phenomenon to resolve the ambiguity.

By mapping the intellectual structure of low-code literature, we seek to contribute to developing a comprehensive understanding of the low-code phenomenon and the diverse research streams that exist within it. The identified research streams include origins of low-code within software engineering, low-code and emerging software engineering trends, enabling new software development practices, low-code methodologies, enhancing understanding about low-code adoption and leveraging low-code for digital transformation. These research streams offer interesting insights into how organizations utilize low-code. Our work brings structure to the cumulative body of knowledge of low-code and can serve as a guide to future researchers, supporting them to position their work within the low-code research landscape.

References

[1] N. Patkar, A. Chis, N. Stulova, and O. Nierstrasz, "Interactive behavior-driven development: a low-code perspective," in *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, IEEE, 2021.

[2] M. Lebens, R. Finnegan, S. Sorsen, and J. Shah, "Rise of the citizen developer," *Muma Business Review*, vol. 5, pp. 101–111, 2021.

[3] M. De Reuver, C. Sørensen, and R. C. Basole, "The digital platform: A research agenda," *Journal of Information Technology*, vol. 33, no. 2, pp. 124–135, 2018.

[4] S. A. A. Naqvi, M. P. Zimmer, R. Syed, and P. Drews, "Understanding the socio-technical aspects of low-code adoption for software development," *European Conference on Information Systems (ECIS)*, 2023.

[5] D. Di Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer, "Low-code development and model-driven engineering: Two sides of the same coin?," *Software and Systems Modeling*, vol. 21, pp. 437–446, 2022.

[6] F. Sufi, "Algorithms in low-code-no-code for research applications: A practical review," *Algorithms*, vol. 16, no. 2, p. 108, 2023.

[7] R. Benac and T. K. Mohd, "Recent trends in software development: Low-code solutions," *Lecture Notes in Networks and Systems*, pp. 525–533, 2021.

[8] A. C. Bock and U. Frank, "Low-code platform," *Business & Information Sys Eng*, vol. 63, pp. 733–740, 2021.

[9] H. D. White and B. C. Griffith, "Author cocitation: A literature measure of intellectual structure," *Journal of the American Society for Information Science*, vol. 32, no. 3, pp. 163–171, 1981.

[10] B. R. Barricelli, F. Cassano, D. Fogli, and A. Piccinno, "End-user development, end-user programming and end-user software engineering: A systematic mapping study," *Journal of Systems and Software*, vol. 149, pp. 101–137, 2019.

[11] M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice," *Synthesis Lectures on Software Engineering*, vol. 3, no. 1, pp. 1–207, 2017.

[12] F. Gürcan and G. & Taentzer, "Using microsoft powerapps, mendix and outsystems in two development scenarios: An experience report," *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS-C)*, 2021.

[13] N. Prinz, C. Rentrop, and M. & Huber, "Low-code development platforms–a literature review," *Americas Conference on Information Systems (AMCIS)*, 2021.

[14] D. Pinho, A. Aguiar, and V. Amaral, "What about the usability in low-code platforms? A systematic literature review," *Journal of Computer Languages*, 2022.

[15] S. Käss, S. Strahringer, and M. Westner, "Drivers and inhibitors of low code development platform adoption," *IEEE 24th Conference on Business Informatics (CBI)*, 2022.

[16] C. Parker, S. Scott, and A. Geddes, "Snowball sampling," *SAGE Research Methods Foundations*, 2019.

[17] M. Bastian, S. Heymann, M. Jacomy, *et al.*, "Gephi: An open source software for exploring and manipulating networks," *International Conference on Web and Social Media*, vol. 8, pp. 361–362, 2009.

[18] S. Martin, W. M. Brown, R. Klavans, and K. W. Boyack, "Openord: an open-source toolbox for large graph layout," in *Visualization and Data Analysis*, vol. 7868, pp. 786–806, International Society for Optics and Photonics, 2011.

[19] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, no. 10, pp. P1–8, 2008.

[20] Y. Luo, P. Liang, C. Wang, M. Shahin, and J. Zhan, "Characteristics and challenges of low-code development: The practitioners' perspective," *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–11, 2021.

[21] J. J. Gibson, "The theory of affordances," *The Ecological Approach to Visual Perception, Hilldale, USA*, vol. 1, no. 2, pp. 67–82, 1977.

[22] J. G. Greeno, "Gibson's affordances," *Psychological Review*, vol. 101, no. 2, pp. 336–342, 1994.

Appendix

Given page length constraints, we offer a comprehensive reference list of the 105 articles included in our study at https://zenodo.org/record/8354060.