Software and Systems Modeling (2022) 21:1959–1981
https://doi.org/10.1007/s10270-021-00964-0

THEME SECTION PAPER

Modelling in low-code development: a multi-vocal systematic review

Alessio Bucaioni[1] · Antonio Cicchetti[1] · Federico Ciccozzi[1]

Received: 19 July 2021 / Revised: 29 September 2021 / Accepted: 30 November 2021 / Published online: 19 January 2022

Abstract
In 2014, a new software development approach started to get a foothold: low-code development. Already from its early days, practitioners in software engineering have been showing a rapidly growing interest in low-code development. In 2021 only, the revenue of low-code development technologies reached 13.8 billion USD. Moreover, the business success of low-code development has been sided by a growing interest from the software engineering research community. The model-driven engineering community has shown a particular interest in low-code development due to certain similarities between the two. In this article, we report on the planning, execution, and results of a multi-vocal systematic review on low-code development, with special focus to its relation to model-driven engineering. The review is intended to provide a structured and comprehensive snapshot of low-code development in its *peak of inflated expectations* technology adoption phase. From an initial set of potentially relevant 720 peer-reviewed publications and 199 grey literature sources, we selected 58 primary studies, which we analysed according to a meticulous data extraction, analysis, and synthesis process. Based on our results, we tend to frame low-code development as a set of methods and/or tools in the context of a broader methodology, often being identified as model-driven engineering.

Keywords Low-code development · Modelling · Model-driven engineering · Systematic review

1 Introduction

Nowadays, software operates virtually all the existing systems in every domain, from banking to avionics. As a consequence, software systems are becoming utterly complex and their development costly and time-consuming. Since its inception, software engineering has focused on mitigating these issues by improving software development processes, methods, and tools at the disposal of engineers.

At the dawn, it was computer-aided software engineering (CASE), which proposed to use computer facilitated tools and methods for mastering the development of large software projects hence improving productivity and reducing error-proneness [26]. Through the years, CASE evolved and laid the foundations for new, adjacent software engineering paradigms such as visual programming [24] and model-driven engineering (MDE) [75].

MDE shook software engineering by shifting the development focus from programming to "modelling", thereby placing abstraction, separation of concerns, and automation at the centre of the development. Modern software development relies on domain-specific abstractions described through domain-specific languages (DSL) [58]. Computer programs—model transformations in the case of domain-specific modelling languages (DSML) [76]—(automatically) manipulate these abstractions. Abstraction and domain-specific conceptualisation given by DS(M)L, together with automation provided by model transformations, help domain experts (who may or may not be software experts) to develop software systems in a domain-focused and human-centric manner.

In 2014, a "new" software development approach started to get a foothold: low-code development (LCD) [67]. LCD can be described as a visual and semi-automated approach to software development. The basic idea is that by using LCD an engineer shall be able to abstract and semi-automate every

Communicated by Esther Guerra.

✉ Antonio Cicchetti
antonio.cicchetti@mdh.se

Alessio Bucaioni
alessio.bucaioni@mdh.se

Federico Ciccozzi
federico.ciccozzi@mdh.se

[1] School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden

step of a software system life-cycle and streamlining the delivery of multiple systems in a given domain. LCD has been positively received by industry and its industrial adoption has been increasing at staggering levels. Gartner Incorporated (Gartner) predicted that by 2024 "*low-code development will be responsible for more than 65% of application development activity*" and that "*75% of large enterprises will be using at least four low-code development tools for both IT application development and citizen development initiatives*"[1].

Following the remarkable industrial adoption, the software engineering research community in general—and the MDE community in particular—started to develop an interest on LCD. As confirmed by this study, LCD and MDE share several core principles such as abstraction, automation, visual notations, and agility. In 2020, the International Conference on Model Driven Engineering Languages and Systems (MODELS) hosted "*LowCode: Low-Code Development Platforms*", the first international workshop on LCD. In 2021, MODELS hosted the second instance of such a workshop highlighting the relevance of LCD for the MDE community and the great potential for synergies between the two.

Given the mentioned premises in both research and practice, it is safe enough to position LCD technology adoption in its *hype of inflated expectations* phase[2]. Therefore, we believe that the time is ripe to create a structured and detailed snapshot of what LCD is considered to be especially concerning modelling and MDE.

In this work, we report on the planning, execution, and results of a multi-vocal systematic review, which provides a structured and comprehensive overview of LCD and its relation to modelling and MDE. The fact that LCD had spread significantly among communities of citizen and professional developers, enterprise architects and information technology leaders highlighted the need for a multi-vocal systematic review [38], to provide a more nuanced look at the topic and side the academic findings with professional practices. From an initial set of 720 peer-reviewed publications and 199 grey sources, we identified 58 *primary* studies, which we analysed thoroughly following a meticulous data extraction, analysis, and synthesis process.

A summary of the highlights resulting from our study is as follows:

- While the term "low-code" was coined in 2014, first peer-reviewed publications on LCD appeared only in 2018. Since then, the publication trend on LCD-related topics has been considerably growing . Nonetheless, almost half of the analysed peer-reviewed studies were published at workshops, indicating LCD research being in a maturation phase.
- MDE is by far the most prominent core technology among the 27 ones considered in our primary studies as important for LCD. Correspondingly, improved return on investment, abstraction, graphic user interface-based development, and automation are the most mentioned benefits linked to the adoption of LCD.
- The grey literature tends to emphasise the rapid prototyping and usability features of LCD solutions, which are expected to disclose trial-and-error ways to modernisation without large upfront investments (as opposed to MDE and other software engineering methods).
- There is an extensive catalogue of 39 tools supporting LCD. In this respect, both peer-reviewed and grey literature focus on solutions for end-users, while the issues related to the creation of new/customised LCD platforms are largely neglected.

The remainder of this article is organised as follows. Section 2 describes the research method in all its phases. Sections 3 and 4 present the results of the vertical and orthogonal analyses, respectively. Section 5 draws relations between LCD and MDE. Section 6 discusses the main threats to validity for this study and related mitigation strategies. Section 7 compares our work with the related literature, while Sect. 8 concludes the article with final remarks and potential future works.

2 Research method

We designed and carried out this study by following the guidelines by Kitchenham and Brereton on systematic reviews in software engineering [51] and complementing them with those for conducting multivocal literature reviews in software engineering by Garousi et al. [38]. Figure 1 depicts the process that we followed, which consisted of three phases being *planning*, *conducting*, and *documenting*.

Planning phase. The main goals of this phase were to:

- Establish the need for a systematic review on LCD,
- Identify the research goal (RG) and related research questions (RQs), and
- Define the research protocol to be followed by the researchers to carry out the study in a systematic manner.

A detailed research protocol was the main output of the planning phase.

Conducting phase. In this phase, we performed all the activities defined in the research protocol, which were *search and*

[1] https://www.mendix.com/resources/gartner-magic-quadrant-for-low-code-/application-platforms/

[2] https://www.gartner.com/en/research/methodologies/gartner-hype-cycle

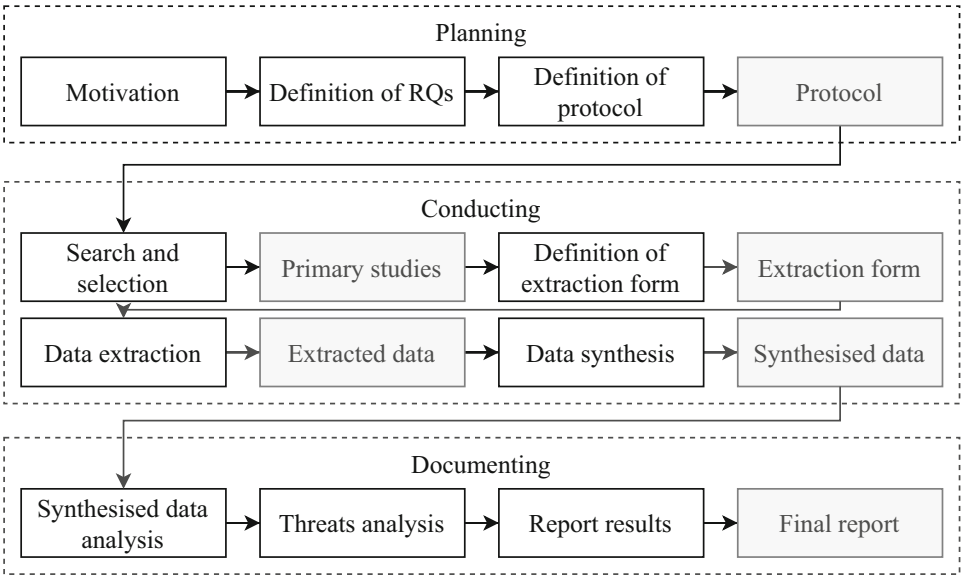


Fig. 1 Overview of the process

selection, *data extraction form definition*, *data extraction*, and *data analysis*.

- We started the search and selection step by performing an automatic search of peer-reviewed literature on a set of four scientific databases and indexing systems. Besides, we conducted an automatic search of the grey literature on the Google search engine. Then, we used selection criteria for filtering the identified candidate entries to obtain the set of primary studies to be included in later activities of the review. We complemented the initial automatic search of with an exhausting backward and forward snowballing (on Google Scholar) as suggested by Wohlin et al. [88].
- In the data extraction form definition, we defined the set of parameters used for classifying and comparing the primary studies. The set was based on the research questions and was built systematically and iteratively using the standard key-wording process [65]. We used the identified parameters for designing the data extraction form, which we used for data extraction activities.
- In the data extraction step, we analysed each of the identified primary studies to fill the data extraction forms. We collected and aggregated the filled forms for data analysis and synthesis.
- In the data analysis step, we analysed the extracted data. The main goal of such an analysis was to provide answers to the research questions. To this end, we performed both quantitative and qualitative analysis using vertical and orthogonal analyses.

Documenting phase. The main goals of this phase were:

- analyse and document possible threats to validity affecting this systematic mapping study, and
- document this study and its results. To support independent replication and verification of our study, we provide a complete and public replication package[3] containing the data from search and selection, the complete list of primary studies, the data extraction forms as well as the scripts used for analysis and synthesis.

2.1 Research goal and questions

We defined the research goal of this systematic review as:

> *identify, classify, and evaluate publication trends, founding concepts, benefits, supporting tools, and business domains of LCD.*

We defined the research goal using the Goal-Question-Metric perspectives [21] as Table 1 illustrates.

following five research questions, which contribute to different and unique objectives of the investigation.

> *RQ1 Which are the publication trends in LCD research?* LCD is a research area with contributions from both industry and academia from several research groups focusing on specific issues with different levels of independence and methodologies and at a different level of detail.

[3] The replication package is available at: https://github.com/federicoCiccozzi/lowCode_replicationPackage/

Table 1 Research goal

Purpose	Identify, classify, and evaluate
issue	publication trends, founding concepts, benefits, supporting tools, and business domains
Object	of low-code development
viewpoint	from the point of view of researchers and practitioners

RQ2 Which are the core technologies supporting LCD? LCD is a relatively new research area, which cross-cuts several other areas of research.
RQ3 Which are the business domains to which LCD is applied? It is not clear to which extent LCD can be successfully applied to different business domains.
RQ4 How does the landscape of tools supporting LCD looks like? The ultimate goal of LCD is the provision of environments for easing the development efforts of software-intensive systems.
RQ5 Which are the main reported benefits of LCD? LCD advocates several benefits ranging from technical to business ones.

Answers to RQ1 provide a detailed snapshot of the set of academic venues where research ideas and results on LCD are presented and discussed, and the density of scientific interest in it. By answering RQ2, we draw relations among LCD and those existing software engineering technologies that can be considered as founding pillars of LCD. By answering RQ3 and RQ4, we provide a comprehensive catalogue of the business domains where LCD has been successfully applied and the tools supporting LCD. Eventually, answers to RQ5 provide a snapshot of the main reported benefits of LCD.

2.2 Search and selection strategy

Following the steps shown in Fig. 2, we collected the set of relevant research studies for our investigation.

We carried out two parallel review activities: *review of the peer-reviewed literature* and *review of the grey literature*. We followed the same overall process for both the review activities. For the sake of simplicity, in the remainder of this article we refer to included studies from either source as primary studies and use different terms only when strictly needed. For the peer-reviewed search, we selected four of the largest and most complete scientific databases and indexing systems in software engineering [51] based on their high accessibility and their reputation as being an effective means for supporting systematic reviews in software engineering [22]. The selected databases and indexing systems are: *IEEE Xplore Digital Library*, *ACM Digital Library*, *SCOPUS*, and *Web of Science* (Table 2).

Starting from our research goal and questions, we created a search string that we used for exercising the databases and indexing systems. Considering that LCD is a relatively new field of research, we decided to use a simple search string that helped us in collecting as many relevant research works as possible. Despite we are knowledgeable of the existence of several nuances in the terminology, notably the alternative use of no-code or low-code terms, our experiences identify low-code as the most generic and inclusive one, while others are used as specialisations of low-code and hence mentioned jointly. We discuss possible threats to validity related to the search string in Sect. 6; the search string is:

"low code"

The automatic search for peer-reviewed literature resulted in a total of *720* potential peer-reviewed studies. From this initial set, we removed impurities (results that were not research papers) and merged duplicates, reaching a new set of 450 potential primary studies.

For the grey literature search, we used the Google search engine (Table 2), which accounts for 92.2% of global web searches.[4] Similarly to what we did for the peer-reviewed search, we created a search string that we used for exercising the search engine. For the grey literature search, we used a string more focused on LCD in combination with MDE for avoiding an overwhelming number of irrelevant results and to complement the peer-reviewed literature search with the most relevant sources for our scope. The string is:

(("model driven" OR "model based") AND "low code")

The search for grey literature, carried out as a combination of automatic and manual activities, resulted in a total of potentially relevant *199* hits. It should be noted that Google performs an automatic similarity check on the displayed results aiming at removing impurities and duplicated entries.

According to the selection process proposed by Ali and Petersen [16], we applied inclusion and exclusion criteria on the search results so to ensure an objective selection. The set of selection criteria used that we used for filtering search results studies was:

– Inclusion criteria for peer-reviewed literature

[4] https://gs.statcounter.com/search-engine-market-share

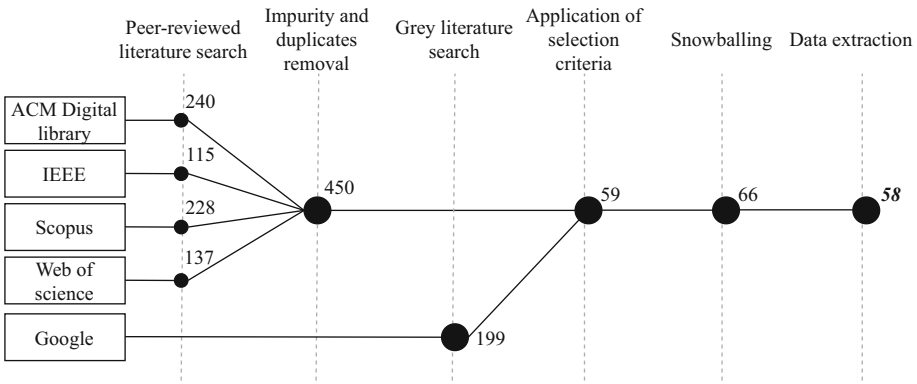


Fig. 2 Overview of the search and selection process

Table 2 Electronic databases, indexing systems, and search engine used in this study

Name	Type	URL
IEEE xplore digital library	Electronic database	http://ieeexplore.ieee.org
ACM digital library	Electronic database	http://dl.acm.org
SCOPUS	Indexing system	http://www.scopus.com
Web of Science	Indexing system	http://webofknowledge.com
Google	Search engine	http://google.com

1. Studies proposing a formalisation, an approach, a use or a tool on LCD.
2. Studies proposing an evaluation of their main contribution.
3. Studies subject to peer review [87].
4. Studies written in English.
5. Studies available as full-text.

– Inclusion criteria for grey literature

1. Websites focusing on low-code development and some aspect of model-based/driven development.
2. Websites in English.

– Exclusion criteria for peer-reviewed literature

1. Secondary and tertiary studies (e.g., systematic literature reviews, surveys, etc.).
2. Studies in the form of tutorial papers, short papers ($\leq$4 pages), poster papers, editorials, manuals, because they do not provide enough information.

– Exclusion criteria for grey literature

1. Websites referring to peer-reviewed literature[5].

To be included in the next step, a study had to meet all the inclusion criteria and none of the exclusion criteria for its category. By applying the selection criteria on the peer-reviewed and grey literature studies, we eliminated 420 peer-reviewed papers and 170 grey literature entries, and reached a set of *59* potential primary studies (*30* peer-reviewed publications and *29* grey literature entries). Then, we performed a closed recursive backward and forward snowballing activity [88], which helped us in minimising potential bias concerning construct validity [40]. As a result, we found 4 additional peer-reviewed studies and 3 grey literature entries, which led us to a total of *66* primary studies (*34* peer-reviewed publications and *32* grey literature entries). Note that for grey literature entries, all exclusions were due to the entry not treating LCD with any MDE-related aspect.

[5] We kept the peer-reviewed and grey literature selection processes separated due to the corresponding guidelines; however, we made sure that any potentially relevant peer-reviewed entry excluded for the grey literature was already considered in the peer-reviewed literature selection.

2.3 Data extraction

To extract and collect data from the primary studies, we created the *data extraction form* shown in Table 3. The data extraction form is composed of five facets, one per research question. RQ1 comprises two clusters. In the first cluster, we included standard information such as title, authors, and publication details of each study. In the second cluster, we included the type of research, following the taxonomy by Wieringa et al. [86], as well as the type of questions, results, and validation, according to the taxonomy by Shaw et al. [78]. Considering that facet RQ1 is only relevant for the peer-reviewed literature, we did not use it when extracting data from the grey literature. For the other facets, RQ2 to RQ5, we carried out a *keywording* systematic process to develop an extraction form that could fit the set of primary studies and take their characteristics into account [65].
First, we collected keywords and concepts by reading the full text of the primary studies. Then, by adopting a process

Table 3 Data extraction form facets, clusters, and categories

Facet	Cluster	Category	Description	Value
RQ1	Publication details	Authors	Identifies the list of authors	String
		Publication title	Identifies the title of the study	String
		Venue name	Identifies the name of the venue	String
		Venue type	Identifies the type of venue	Workshop, conference, journal, book chapter
		Year	Identifies the year of publication	Numeric value (e.g., 2020)
	Publication analysis	Research type	Identifies the type of the research according to the taxonomy in [86]	Evaluation research, proposal of solution, validation research, philosophical paper, opinion paper, experience paper, survey paper
		Questions type	Identifies the type of research questions of the study according to the taxonomy in [78]	Method or means of development, method for analysis, design/evaluation or analysis of a particular instance, generalisation or characterisation, feasibility
		Results type	Identifies the type of results of the study according to the taxonomy in [78]	Procedure or technique, qualitative or descriptive model, empirical model, analytic model, notation or tool, specific solution, answer or judgement, report
		Validation type	Identifies the type of validation of the study according to the taxonomy in [78]	Analysis, experience, example, evaluation, persuasion, blatant assertion
RQ2	—	Core technologies	List of core technologies as identified in the studies	String
RQ3	—	Domains	List of business domains as identified in the studies	String
RQ4	—	Tools	List of tools as identified in the studies	String
RQ5	—	Benefits	List of benefits as identified in the studies	String

similar to the sorting phase of the grounded theory methodology [27], we clustered the elicited keywords and concepts into categories. When we collected further information that was deemed relevant, but that was not captured by the current extraction form, we reviewed the additional information and, if needed, refined the data extraction form. Eventually, we re-analysed the primary studies according to the refined form and extracted new data. During the data extraction phase, we excluded additional 8 peer-reviewed studies leaving us to a final set of *58* primary studies. The studies were excluded after screening their full texts and deeming them as irrelevant to this research. Tables 4 and 5 list the final set of primary

Table 4 List of peer-reviewed primary studies

ID	Title	Author	Year	References
P1	App Development via Low-Code Programming as Part of Modern Industrial Engineering Education	Adrian et al.	2020	[13]
P2	Improving the developer experience with a low-code process modelling language	Henriques et al.	2018	[42]
P3	Low-code platform for automating business processes in manufacturing	Waszkowski	2019	[84]
P4	RESTsec: a low-code platform for generating secure by design enterprise services	Zolotas et al.	2018	[90]
P5	Towards a seamless integration of iot devices with iot platforms using a low-code approach	Pantelimon et al.	2019	[63]
P6	Towards Automating the Construction of Recommender Systems for Low-Code Development Platforms	Almonte et al.	2020	[18]
P7	Democratizing the Development of Recommender Systems by Means of Low-Code Platforms	Di Sipio et al.	2020	[32]
P8	Low-Code Engineering for Internet of Things: A State of Research	Ihirwe et al.	2020	[44]
P9	A strategy for facing new employability trends using a low-code development platform	Metrolho et al.	2020	[62]
P10	Low-code as enabler of digital transformation in manufacturing industry	Sanchis et al.	2020	[74]
P11	Supporting the understanding and comparison of low-code development platforms	Sahay et al.	2020	[72]
P12	Eciently Querying Large-Scale Heterogeneous Models	Ali et al.	2020	[17]
P13	Towards Access Control for Collaborative Modelling Apps	Brunschwig et al.	2020	[23]
P14	DevOpsML: Towards Modeling DevOps Processes and Platforms	Colantoni et al.	2020	[30]
P15	Development and Validation of a Descriptive Cognitive Model for Predicting Usability Issues in a Low-Code Development Platform	Silva et al.	2020	[79]
P16	An overview on how to develop a low-code application using OutSystems	Martins et al.	2020	[59]
P17	BPMN and DMN for Easy Customizing of Manufacturing Execution Systems	Peinl et al.	2019	[64]
P18	Intellectual Technologies in Digital Transformation	Sakhnyuk et al.	2020	[73]
P19	Lowcomote: Training the next generation of experts in scalable low-code engineering platforms	Tisi et al.	2019	[81]
P20	Sagitec Software Studio (S3) - A Low Code Application Development Platform	Arora et al.	2020	[20]
P21	A Natural Language Driven Approach for Automated Web API Development: Gherkin2OAS	Dimanidis et al.	2018	[33]
P22	Towards a Low-Code Solution for Monitoring Machine Learning Model Performance	Kourouklidis et al.	2020	[54]
P23	An Approach using a Low-Code Platform for Retraining Professionals to ICT	Metrolho	2019	[61]
P24	Towards transparent combination of model management execution strategies for low-code development platforms	Philippe et al.	2020	[66]
P25	Towards the next generation of reactive model transformations on low-code platforms: three research lines	Horvath et al.	2020	[43]
P26	Challenges and opportunities in low-code testing	Khorram et al.	2020	[50]

Table 5 List of grey primary studies

ID	Title	Author	Year	References
G1	Low-Code Principle #1: Model-Driven Development, The Most Important Concept in Low-Code	Johan den Haan	2020	[46]
G2	Introducing the Low-Code Manifesto	Johan den Haan	2021	[47]
G3	Low-code. High impact.			[5]
G4	Low Code Applications and Model-Driven Development	Srdjan Kovacevic	2020	[80]
G5	Do Model-Driven Development (MDD) platforms make good on their promises?	Jaco van Kooten	2016	[45]
G6	Get the right mix of low-code and pro code to modernize your enterprise applications	Christoph Garms	2020	[28]
G7	Introducing model-driven apps—a new way to create	Adrian Orth	2018	[14]
G8	5 FAQs on the low-code approach	Ryan Black	2020	[71]
G9	Low Code / No Code Development Platform			[3]
G10	What will Power Fx mean for Model-driven Power Apps?	Jukka Niiranen	2021	[49]
G11	Genio			[2]
G12	Evaluating low-code platforms: what's under the hood?	Erlend Barstad Strand		[34]
G13	Top Evaluation Criteria for Selecting a Low-code Platform— Part 2	Maggie Burnham	2021	[57]
G14	Low-Code Support			[56]
G15	Model Driven Development vs. Code Generation. What's the right balance?	Evelina Brown	2020	[36]
G16	Process Meets Intelligence: From Workow To Automated Enterprise	Johan den Haan	2021	[48]
G17	What is it like to use J2Paas low-code platform to develop software?			[10]
G18	Low-code without low-code limitations	Johan den Haan	2021	[6]
G19	Mendix Versus OutSystems—Make an Informed Decision			[7]
G20	Siemens Digital Industries Software			[8]
G21	Futureproof your business with lasting agility			[1]
G22	Siemens PLM Becomes Siemens Digital Industries Software	Rob Spiegel	2019	[68]
G23	Low-Code / No-Code in an enterprise landscape	Harald van der Weel	2017	[41]
G24	What Is Low-Code?	Adrian Peng	2021	[15]
G25	Yida Plus, Alibaba's Solution for Low-Code Development	Zhao Yuanling	2019	[89]
G26		Verkstads Forum	2021	[82]
G27	Guide to choose a low-code development technology	Sandra Melo	2021	[82]
G28	CEnterprise software and Application Services. Extract-Digital Enablement Via Low Code Platforms: Understanding The Position, Uncovering The Prospects	Angela Eager	2019	[19]
G29	No-code/low-code: Why you should be paying attention	Setrag Khoshafian	2021	[77]
G30	Low-Code and UX: Can They Work Together?			[4]
G31	What is Low-Code Automation and How Could it Benet You?	Claire Vanner	2021	[29]
G32	WEM: No-Code Development to Democratize Cloud-Based Application Functionality Creation	George Mironescu	2018	[39]

studies, grouped by source type (peer-reviewed literature and grey literature respectively.

2.4 Data analysis and synthesis

We gathered, analysed, and synthesised the extracted data to understand and classify the current state of the art in the area of LCD [52] using the guidelines presented by Cruzes et al. [31]. In this research, we performed both quantitative and qualitative analyses using vertical and orthogonal analysis techniques. For both vertical and orthogonal analyses, we combined content analysis [37] and narrative synthesis [69]. The combined analysis helped us in categorising and coding approaches under broad thematic categories. The analysis technique provided for a detailed explanation and interpretation of the findings coming from the former analysis. We used vertical analysis for finding trends and collecting information on each facet of the data extraction form according to the so-called line of argument synthesis process [87]. Hence, we first analysed each primary study individually for classifying its main features according to the parameters in the data extraction form. Later, we analysed the whole set of primary studies to uncover and reason about potential patterns. We used orthogonal analysis for identifying possible relations among different facets in the data extraction form. To this end, we grouped and cross-tabulated extracted data and compared pairs of facets of the data extraction form. We used contingency tables for extracting and evaluating relevant pair-wise relations.

3 Results: vertical analysis

Using the data extraction form, we analysed each primary study individually to classify its main features. Later, we analysed the whole set of primary studies as a whole to uncover potential patterns, trends, and gaps. Hereafter, we report and discuss these patterns, trends, and gaps. For each category of the extraction form, we extracted multiple values from the studies. Alternatively, we did not extract any value if none was available. Hence, the total number of occurrences in the plots may differ from the total number of primary studies.

3.1 Publication trends (RQ1)

By answering this research question, we aim at identifying publication trends in terms of the trend over time, venue type, and research type. According to the collected data, research on LCD can still be considered in its embryonic phase.

Despite the first formalisation of LCD dates back to 2014, the first 3 peer-reviewed studies appeared only in 2018 (Fig. 3). LCD is therefore a new research topic. However,

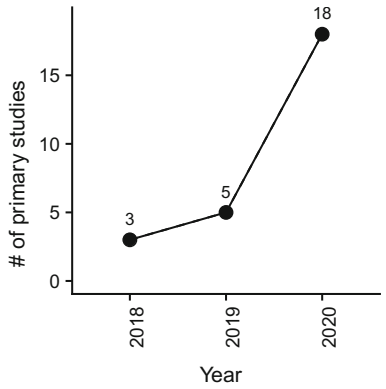


Fig. 3 Publication trend over time

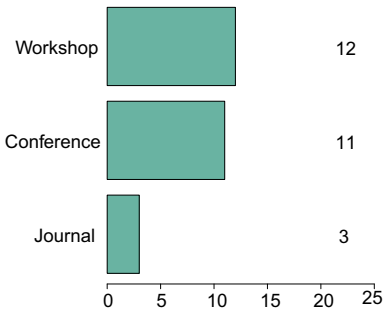
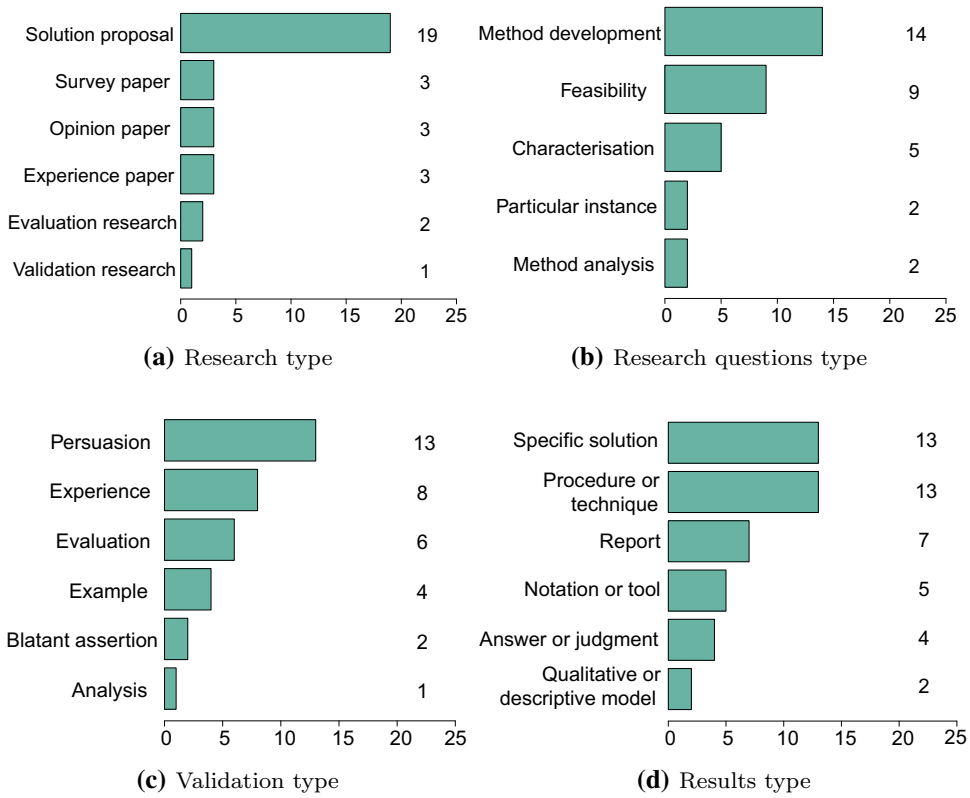


Fig. 4 Publication trend over venue

the interest of the research community is growing considerably. In fact, in 2019 and 2020 there is a 2- and 15-fold increase in the number of included peer-reviewed studies. Considering the steep increment of peer-reviewed studies in 2019 and 2020 and the creation of specific venues such as the "*LowCode: Low-Code Development Platforms*" workshop at MODELS, we expect this trend to continue in next few years.

Most of the peer-reviewed studies (88.4%) were published in either workshops or conferences (Fig. 4). Workshop is the most common venue type and it accounts for 46.2% of the peer-reviewed primary studies, while Conference stands for 42.3%. Journal papers represent 11.5% of peer-reviewed primary studies. The collected data on the venue type seem to confirm our observation about research on LCD being in its embryonic phase. Workshops are commonly the best fitting venues for early results in new research areas. The top venue in terms of number of peer-reviewed studies is the MODELS conference together with its satellite events (e.g., workshops), which accounts for the 42.3% of the identified studies. Besides MODELS, the research community seems not to favour other specific venues and the remaining 57.7% of the identified studies are scattered across 15 different venues. The most common type of research among the peer-reviewed studies is *solution proposal*, which accounts for 61.2% of the studies (Fig. 5a).



(a) Research type **(b)** Research questions type

(c) Validation type **(d)** Results type

Fig. 5 Research, questions, validation, and results types of the peer-reviewed primary studies

Hence, these papers discuss the adoption of LCD for building specific solutions to given problems like integration of internet of things (IoT) devices (P5 [63]) or generation of secure by design enterprise services (P4 [90]). Concerning the type of research questions, method of development (43.7%) and feasibility (28.1%) are the two most common ones. This means that most of the peer-reviewed studies either use LCD for creating methods and solution (P12 [17]) or investigate whether LCD can be used for, e.g., democratizing the development of recommender systems (P7 [32]). In line with research and research question types, the two top-ranked types of results are specific solution (29.5%) and procedure/technique (29.5%). Research, questions and results types show that current research in LCD mainly focuses on solving practical issues. The two top-ranked types of validation are persuasion (38.2%) and experience (23.5%). While these validation types can suffice for some specific research questions, they highlight a lack of rigorous validation in this research area. We believe that this might be partially explained considering by LCD research being at an early stage.

Highlights—RQ1 Publication trends

- Earliest published works on LCD have appeared in 2018.
- The 15-fold increase in peer-reviewed primary studies in 2020 with respect 2018 suggests that the interest of the community on LCD is growing considerably.
- Apart the set of publications at a dedicated MODELS workshop, the dispersed distribution of studies across venues suggests that the research community does not strongly favour any specific one. This suggests the possible appearance of an *ad-hoc* venue in the near future, should the community continue growing.
- Research on LCD is still at its initial phase, thereby proposed solutions have not undergone thorough rigorous validation processes.

3.2 Core technologies (RQ2)

By answering this research question, we aim at identifying the core technologies supporting LCD. According to the collected data, there are 27 technologies reported as pivotal for LCD. Such technologies were either mentioned explicitly in the primary studies or we could infer them from the contents of the studies. Figure 6 shows technologies that were mentioned in at least three primary studies.

MDE is the most mentioned core technology (22.6%). Behind MDE, we find visual programming (13.28%) and component-based (10.9%). We find the three most-mentioned

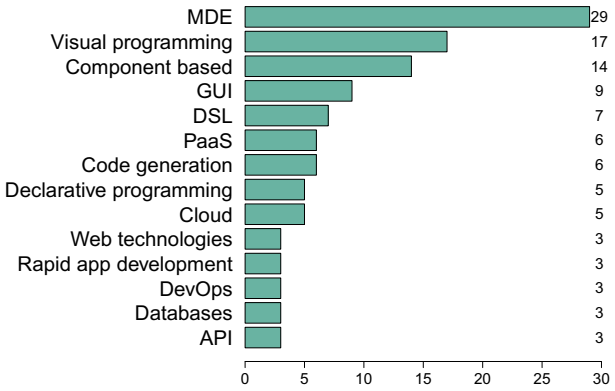


Fig. 6 Core technologies

technologies to be in line with the LCD mission, that is to abstract and automate every step of the software development life-cycle through the use of pre-built visual components. Another set of core technologies is composed by graphic user interface (GUI) (7%), domain-specific language (DSL) (5.4%), platform as a service (PaaS) and code generation (4.6% each), declarative programming and cloud (3.9% each). The usefulness of these technologies for LCD does not come as a surprise as most of the LCD platforms are web-based platforms leveraging domain-specific visual languages. Most of these platforms are delivered using the PaaS model. Web technologies, rapid app development, DevOps, database, and application programming interface (API) are additional technologies mentioned 2.3% of the studies each. Eventually, other less reported technologies are: document generation, Business Process Model and Notation (BPMN), eXtensible Access Control Markup Language (XACML), Software Product Line (SPL), self-service functionality, Systems Applications and Product (SAP), extreme programming, executable modelling, data models, CASE, Artificial Intelligence (AI), access control scheme, and Attribute-Based Access Control (ABAC).

According to our investigation, 22 of the 27 elicited technologies were found in both the peer-reviewed and grey primary studies while 5 of them were found in the grey literature only. These technologies are: API (G6 [28], G12 [34], G18 [6]), SPL (G11 [2]), extreme programming (G10 [49]), executable modelling (G1 [46], G4 [80]), and AI (G11 [2]). According to our investigation, there is a consensus between the data extracted from the peer-reviewed and the grey primary studies, as shown in Fig. 7a, b. One difference that emerges from the comparison between the data from the peer-reviewed and the grey studies is that the latter identify a much smaller set of key technologies (composed of only 5 of them). In contrast, the peer-reviewed studies seem to single out a broader set of 10 technologies.

Highlights—RQ2 Founding paradigms

- There are 27 core technologies supporting LCD. 22 of them were found in both peer-reviewed and grey studies, while 5 of them were found in the grey literature only.
- MDE is the most mentioned technology followed by visual programming and component-based.
- Grey primary studies identify a narrower set of core technologies as compared to peer-reviewed primary studies.

3.3 Domains (RQ3)

By answering this research question, we provide a catalogue of the business domains where research on LCD has been applied to.

According to the elicited data, LCD is cross-domain within the area of software development and applied to 21 different business domains, ranging from web and app development to aeronautics. The business domains were either mentioned explicitly in the primary studies or could be inferred by analysing the context of the described research. Figure 8 shows the domains that were mentioned in at least three primary studies. The top four business domains of LCD are web (16.6%), mobile (15%), enterprise services, and business process (8.3% each). These data seem to be in line with most of the use case reports from LCD platform vendors, which report LCD to be commonly employed for realising or updating software systems in domains, e.g., web, mobile, microservices, IoT, etc. [60]. Besides, the two most cited domains confirm the intrinsic relation between LCD and web development as observed in RQ2. Right behind the top four domains, we find a cluster of other four domains: IoT (6,6%), healthcare (G3 [5], P20 [20]), education (P1 [13], G1 [89]), and databases (5% each). Other business domains include: request handling (P8 [44]), recommender systems (P6 [18], P7 [54]), manufacturing (P10 [74], G25 [89]), industrial training (P1 [13]), DSL engineering (P13 [23]), social media (P24 [66]), process (G32 [66], P8 [44]), marketing (G9 [3]), desktop (G18 [6]), blockchain (P26 [50], G29 [77]), automotive (P24 [66]), AI and aeronautics (P24 [66]).

According to the elicited data, 19 domains were found in both peer-reviewed and grey primary studies while 2 domains, desktop and marketing, were found in grey literature only. Peer-reviewed primary studies seem to focus more on specific business domains while grey studies tend to approach LCD from a more general point of view. Hence, only few grey studies explicitly mention a domain, as shown in Fig. 9b. Similar to RQ2, grey studies identified a much smaller catalogue of domains and only 2 of them were cited by more than 3 studies.

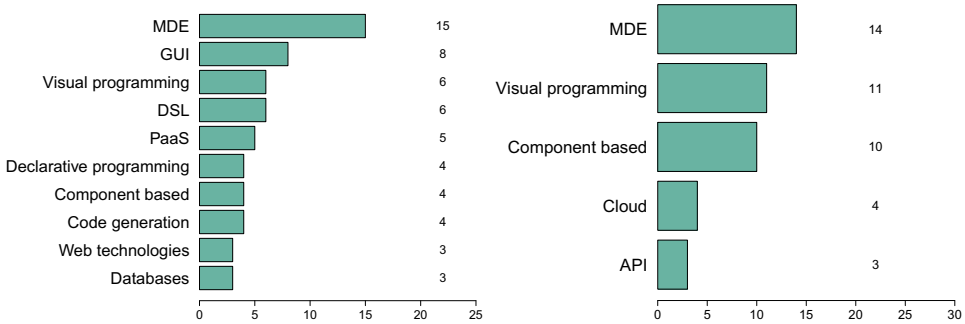


Fig. 7 Comparison between core technologies from the peer-reviewed and grey primary studies

(a) Core technologies from the peer-reviewed primary studies

(b) Core technologies from the grey primary studies

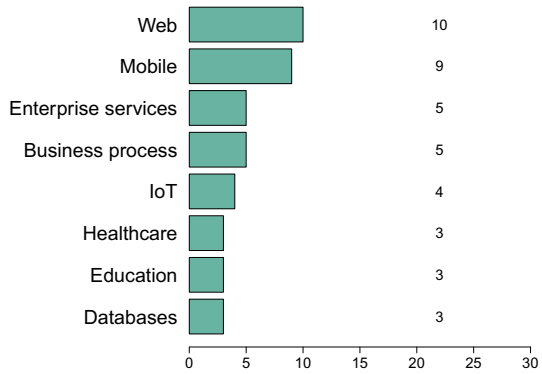


Fig. 8 Domains

Highlights—RQ3 Domains

- In contrast to the wide-spread opinion that LCD is only applicable to few business domains, we found research on LCD to be cross-domain, with a great variation between domain types, from entertainment to safety-critical systems.
- The top four domains targeted by research on LCD are: web, mobile, enterprises services, and business process.
- Interestingly, safety-critical domains such as automotive and aeronautics are also included in LCD research.
- Practitioners seem to perceive LCD as domain-independent and do not explicitly focus on specific domains.

3.4 Tools (RQ4)

By answering this research question, we provide a catalogue of tools supporting LCD and involved in research activities. All the identified tools were explicitly mentioned in the primary studies. According to our investigation, there is an extensive catalogue of 39 tools supporting LCD. Figure 10 shows those tools that were mentioned by at least three primary studies.

The three most mentioned tools are Mendix (16.4%), OutSystems (15.29%), and Microsoft (MS) Power Apps (9.4%). The collected data seem to confirm the reputation of these tools of being the most mature and complete platforms supporting LCD. In 2020, Gartner evaluated these tools as executing well against their current vision and understanding the future market directions [83]. Behind Mendix, OutSystems, and MS Power Apps, we find a cluster of four additional tools: Salesforce and Google App Maker (4,7% each), and Zoho Creator and Appian (3,5% each). Recently, Google App Maker had been discontinued and replaced by AppSheet. While Gartner evaluated Salesforce and Appian as leaders and visionaries, it assessed Zoho Creator as more of a niche player. Surprisingly, Gartner did not include Google App Maker in its evaluation although Google App Maker is rapidly gaining traction, both in research and practice. While the first seven tools account for 57% of the primary studies, the remaining 43% of the primary studies mention a constellation of 32 different tools: Temenos Quantum (P26 [50]), RESTsec (P4 [90]), Pega (G23 [41]) and MS Azure (P18 [73]) (2,3% each) and ZappDev (P12 [17]), TimeSeries (G13 [57]), Sysdev Kalipso (P15 [79]), Simplifier (P8 [44]), Siemens MindSphere (P10 [74]), Sagitec S3 (P20 [20]), PTCThingWork (P10 [74]), Node RED (P8 [44]), Nintex Workflow Cloud (P15 [79]), MIT App Inventor (P15 [79]), Lightening (P26 [50]), Lansa (G18 [73]), Kony (P23 [61]), KissFlow (P16 [59]), INTELLIT (P10 [74]), IDM Cloud (P10 [74]), HiCuMES (P17 [64]), Genio (G11 [2]), Cyclr (G9 [3]), ColdFusion Builder (G9 [3]), Camunda (P17 [64]), Axonivy (G14 [56]), AutoML (P18 [73]), Aurea (P3 [84]), AtmosphericIoT (P8 [44]), App Cloud (P11 [72]), AgilePoint (P20 [20]), and ADAMOS (P10 [74]) (1.1% each).

According to our investigation, the analysed studies tend to leverage LCD tools rather than enhance them although the tooling ecosystem around them changes and it may be customised to achieve specific goals. Pega, TimeSeries, Lansa, Genio, Cyclr, ColdFusion Builder, and Axonivy were only mentioned in the grey studies while the remaining tools were found both in the peer-reviewed and grey literature. While

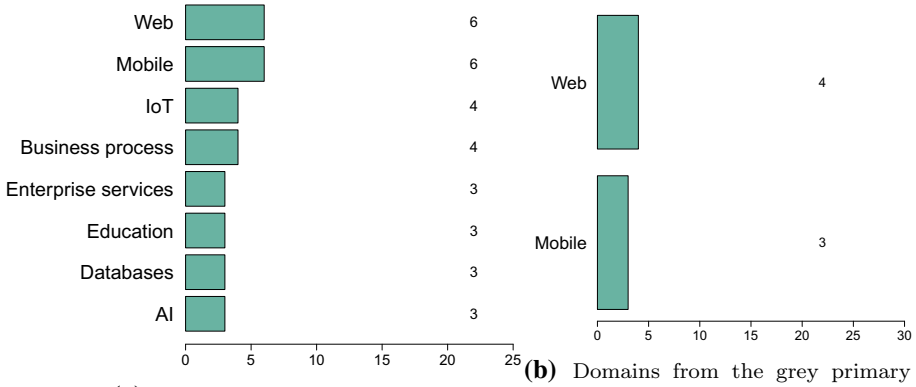


Fig. 9 Comparison between domains from the peer-reviewed and grey primary studies

(a) Domains from the peer-reviewed primary studies

(b) Domains from the grey primary studies

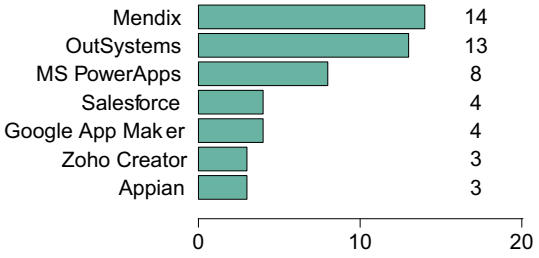


Fig. 10 Tools

grey studies identify a smaller set of tools, both peer-reviewed and grey studies identify Mendix, OutSystems, and MS PowerApps as the top-three leading LCD platforms.

Highlights—RQ4 Tools

- The catalogue of tools supporting LCD is extensive with a total of 39 tools.
- The top three mentioned tools are Mendix, OutSystems, and MS Power Apps.
- LCD tools are leveraged rather than enhanced; the tooling ecosystem around them is customised to achieve specific goals.

3.5 Benefits (RQ5)

By answering this research question, we identify the potential benefits that LCD can bring to software development. The primary studies mentioned explicitly a total of 17 benefits. Figure 12 shows the benefits that were mentioned by at least three primary studies.

Although the distribution of benefits across primary studies is rather disperse, there is a clear gap between the top three benefits and the remaining ones. The top three benefits are improved return of investment (RoI) (19.9%), abstraction (16.36%), and GUI-based development (12.7%). Abstraction is not only one of the most reported LCD benefits, but one of the pillars and benefits of MDE [75]. Improved RoI is often considered as another benefit of MDE directly related to the potential automation introduced by model transformations [76]. No coincidence, MDE was reported as the most important enabling technology for LCD. Another interesting correlation between LCD benefits and core technologies is the one between GUI-based development and GUI. Behind the top three benefits, we have a cluster of four benefits: automation (8.1%), interoperability (6.3%), usability and flexibility (5.4% each). Eventually, we have a further cluster of 6 benefits being: privacy or security and customisability (4.5% each), maintainability (3.6%), reusability, openness and digitisation of business processes (2.7% each). Although reusability is a core quality for several of the reported LCD core technologies (e.g., MDE, component-based), it is not perceived as a top benefit introduced by LCD and only mentioned in few studies. Most of the benefits, 15 out of 17, were found in both peer-reviewed and grey primary studies, while two (openness and scalability) were found in grey studies only. According to the collected data, peer-reviewed studies seem to mostly identify benefits related to the LCD core technologies while the grey studies seem to focus on more practical benefits such as improved RoI, openness, and flexibility.

Highlights—RQ5 Benefits

- There are 17 reported benefits of LCD.
- Improved RoI, abstraction, and GUI-based development are the three most reported benefits.
- Reusability is surprisingly not mentioned as a top benefit.

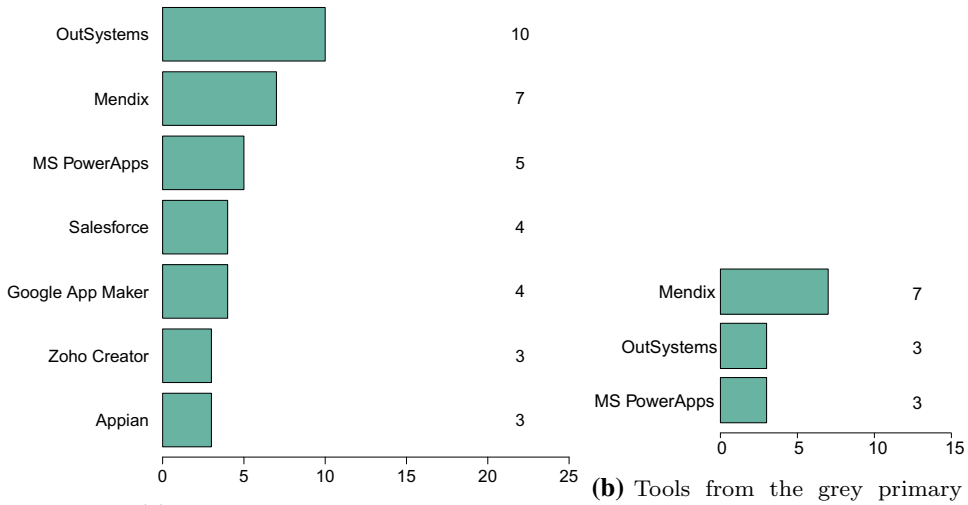


Fig. 11 Comparison between tools from the peer-reviewed and grey primary studies

(a) Tools from the peer-reviewed primary studies

(b) Tools from the grey primary studies

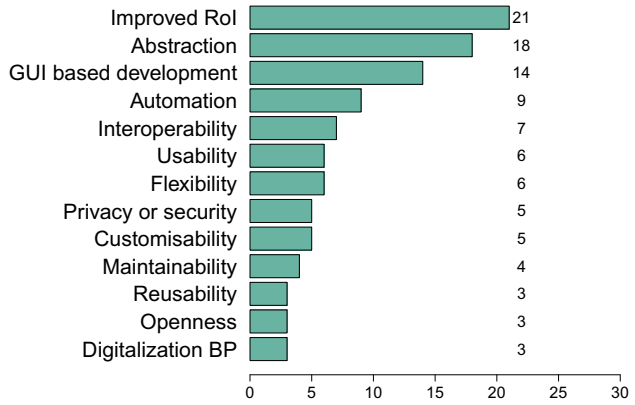


Fig. 12 Benefits

4 Results: orthogonal analysis

In this section, we highlight interesting pair-wise correlations between categories of the extracted data. We carried out this analysis by cross-tabulating and grouping the extracted data to then compare pairs of categories. Eventually, we extracted and evaluated only relevant pair-wise correlations through contingency tables (represented by bubble charts in the section).

4.1 Questions type versus results type

In Sect. 3, we mentioned that the most common types of research questions for peer-review primary studies are method development and feasibility. Similarly, we found out that the most common types of results are specific solution and procedure or technique. The correlation analysis between the types of questions and results shows that there is a rather precise balance between their combinations (Fig. 14).

Such a correlation analysis confirms that research on LCD mostly focuses on development and assessing the feasibility of new or improved specific solutions.

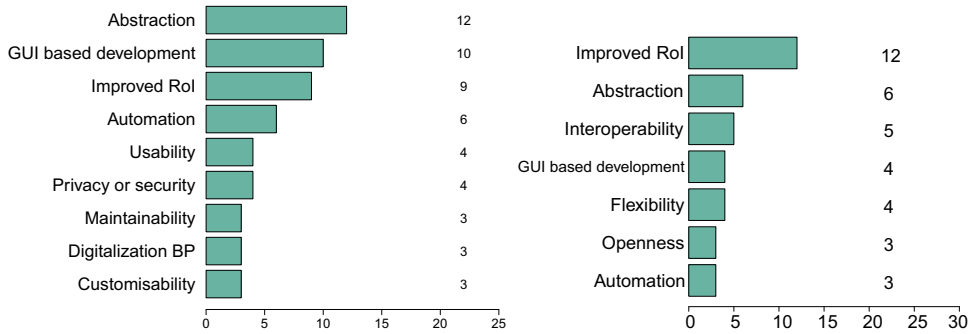
4.2 Questions type versus validation type

The vertical analysis on the type of validation of the peer-reviewed studies highlighted that persuasion and experience are the two most common types of validation techniques. The correlation analysis between the type of questions and validation shows a strong relation between method development and experience, and between method development, feasibility, and persuasion (Fig. 14).

Experience is mostly used for validating method development type of questions, while persuasion is equally used for both method development and feasibility types. It is interesting to note that validation by persuasion suggests that the authors did not focus on an in-depth validation, but rather '*persuade*' readers using their arguments and reasoning. A possible reason for this is that the type of peer-reviewed studies accounted for in this paper were rather preliminary and focusing on pretty new technologies or combinations of them. An example is the work by Philippe et al. (P24) describing initial ideas and a preliminary proof-of-concept on the transparent combination of model management execution strategies for LCD [66].

4.3 Core technologies versus benefits

In the previous section, we found out that LCD revolves around 27 core technologies (or foundations) and that it is



(a) Benefits from the peer-reviewed studies **(b)** Benefits from the grey studies

Fig. 13 Comparison between benefits from the peer-reviewed and grey studies

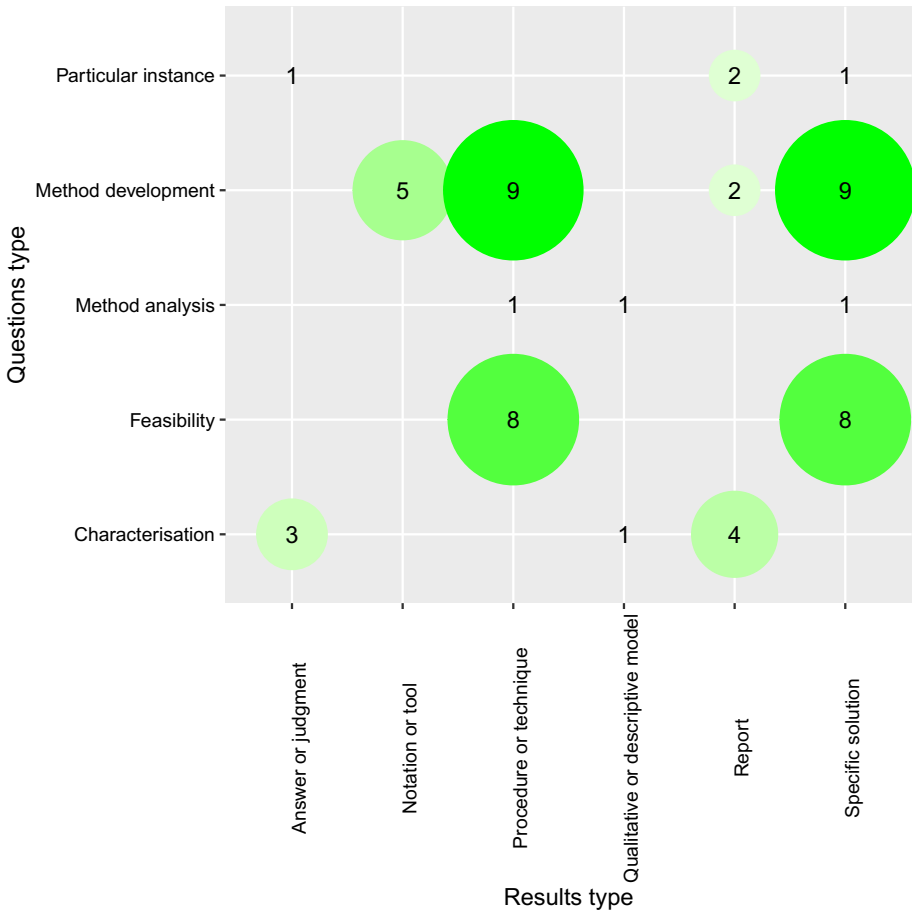


Fig. 14 Questions type versus results type

associated to 17 benefits. Analysing the correlation between core technologies and benefits, we found out that the two most common benefits, improved RoI and abstraction, appear together with two of the three most reported core technologies, MDE and component based.

This tends to confirm the rather strong relationship between LCD and MDE (and related paradigms), which appear to share the same foundations. The strong relationships between LCD and MDE are confirmed by other indications such as the fair number of occurrences of MDE in combination with automation and GUI-based development. Some examples of such tight relations can be found in three studies combining the aforementioned foundations and benefits (P3 [84], P19 [81], P22 [54]). Other interesting relationships are those among visual programming, and component based and abstraction, GUI-based development, and improved RoI. These come with no surprise as LCD is often described as "*a visual approach to software development*" [9] based on pre-developed components.

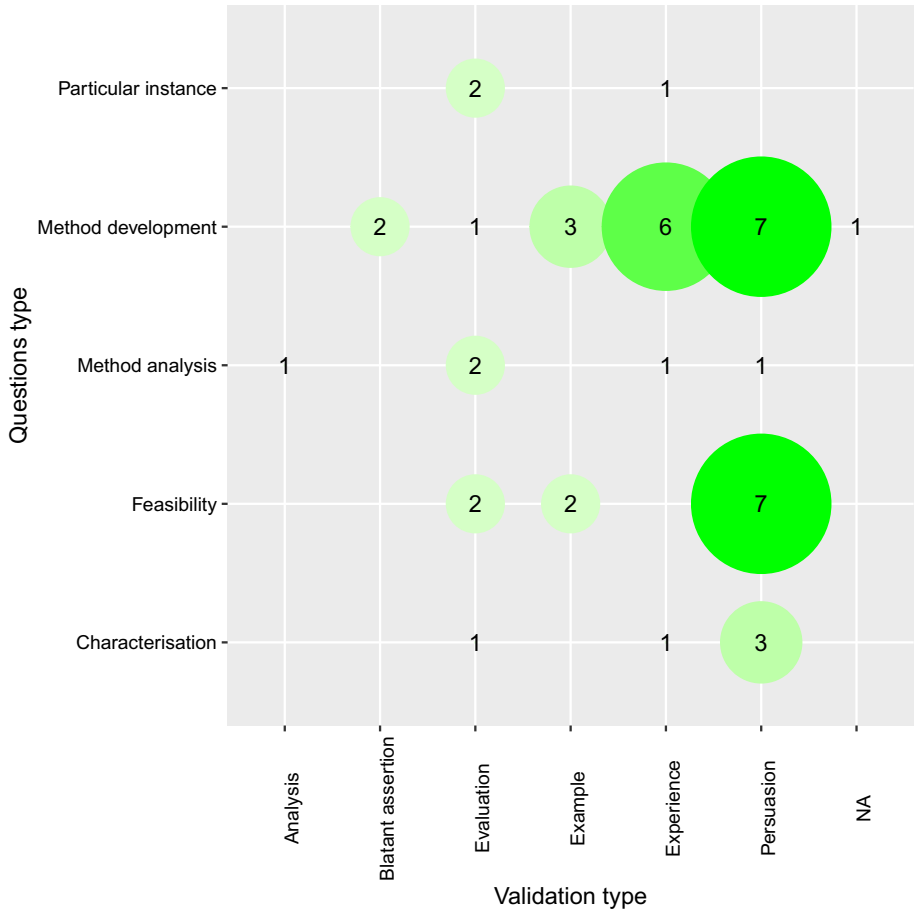


Fig. 15 Questions type versus validation type

4.4 Core technologies versus domains

When analysing the possible correlation between foundations and domain we noticed that MDE is spread across the 90% of the domains represented in our primary studies. This is an additional confirmation that MDE is representative of LCD as core foundation across the business domains, from manufacturing (P3 [84]) to social media (P25 [43]) or blockchain-related technologies (P26 [50]). It is interesting to highlight the fact that GUI and visual programming as foundations have a fair presence across multiple domains, too.

Highlights—Orthogonal analysis

- Research in LCD focuses on improving development and assessing the feasibility of new/improved solutions.
- There is strong relationship between LCD and MDE, that share the same core founding pillars.
- MDE, GUI-based development, and declarative programming are cross-domain foundations of LCD.

5 Reasoning on LCD and its relation to MDE

By reading the most common principles and claims about LCD, it becomes quickly evident that motivations, aims, and technical solutions largely overlap with MDE. Indeed, it is widely accepted in the MDE community to consider LCD as a some sort of synonym for MDE, or to consider MDE techniques as foundations to LCD solutions [25]. As a matter of fact, several MDE-related techniques resemble LCD: notably, there exists a line of research on *by-example* approaches, where activities like metamodelling, model transformation engineering, and model versioning are supported by generative techniques [11]. These techniques are meant to simplify the work of the expert by automatically deriving concepts and relationships, transformation rules, and element matches configurations, respectively. Other solutions include the generation of modelling editors, like the default tree-based editor featured by the Eclipse Modeling Framework and the more advanced annotation mechanism exploited by Eugenia to generate custom concrete syntax [53]. The data extracted and analysed in this review confirms the close relationship between MDE and LCD, and, especially in the grey literature, model-driven and higher-

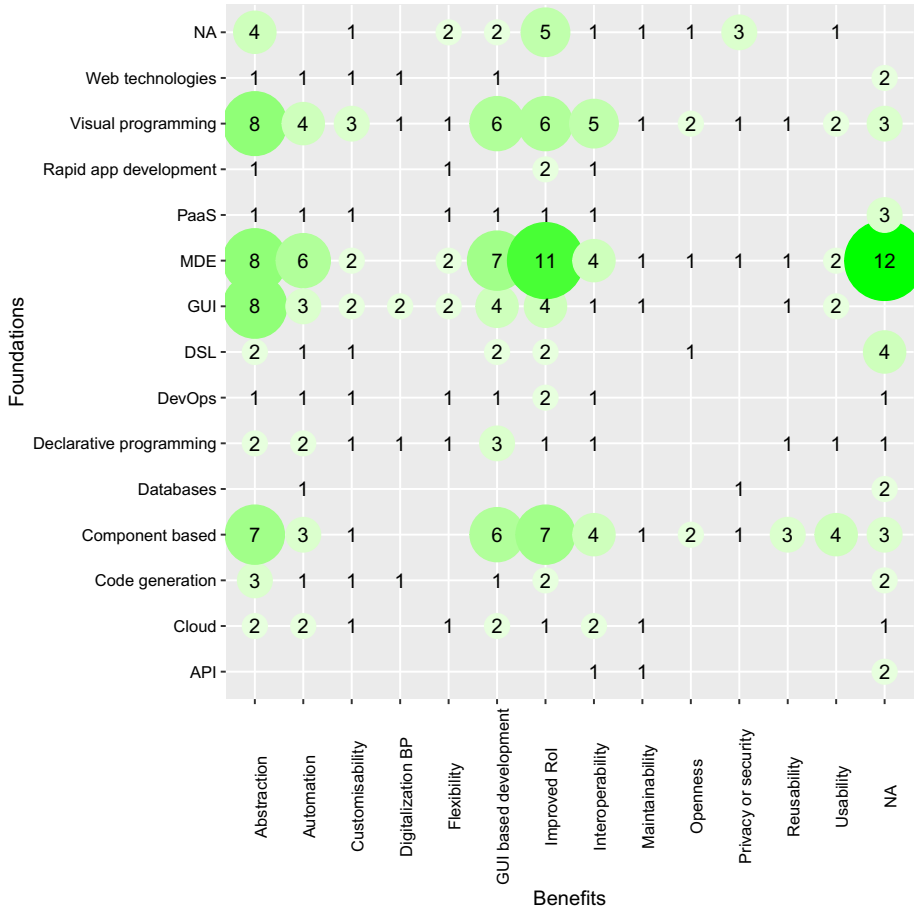


Fig. 16 Core technologies versus benefits

level of abstraction approaches are frequently mentioned as an intrinsic characteristic of LCD.

Some of the analysed literature goes even further, by defining LCD as a maturation step of MDE, especially in terms of usability and flexibility. This aspect is highlighted by the intended peculiarities of LCD: it is a development approach targeting *citizen developers*, which are users with little or no programming experience; it is not required a major effort upfront in order to bootstrap the automation, only little coding refinements might be needed. Indeed, several business-related entries consider LCD as a tool to promote digital transition by a trial-and-error approach: several solutions can be rapidly developed, deployed, tested, and eventually either abandoned for something different or adopted. Moreover, LCD is expected to be more resilient to technological advances, since the frameworks shall abstract away the implementation details from the pre-built components used for a certain application. When comparing these peculiarities to the current state of the practice in MDE, it is possible to highlight what LCD users find as distinctive: even if MDE targeted the reduction of complexity and proposed raising the level of abstraction for enabling domain experts to deal with software design, empirical research testifies that IT literacy is required and that there exist several issues related to tools usability, flexibility, and maintenance [55,85].

In one of its internal initiative, the International Council on Systems Engineering (INCOSE) proposes a catalogue of Model-Based System Engineering methodologies [35]. In that report, a methodology is defined as a recipe to deploy a process, through methods and tools, given certain environmental conditions. Consistently to this definition, the literature reviewed in this article points to LCD as being *a set of methods and/or tools in the context of a broader methodology, being in this case MDE*. In particular, LCD is intended to support citizen developers in very specific tasks, like the configuration of parameters for a machine learning algorithm, the set-up of a business process management system, and so forth. Here, it is worth noticing that the initial definition of *citizen developers* seems to have evolved: they are not supposed to be necessarily users with low to no knowledge in programming, but also, and often, domain experts that need ready-made solutions for specific aspects of their methodology. In this respect, an analogy could be made with model-based web engineering methodologies, for which there exist methods and tools for the automated generation of user interfaces, data management, analytics, etc. [70].

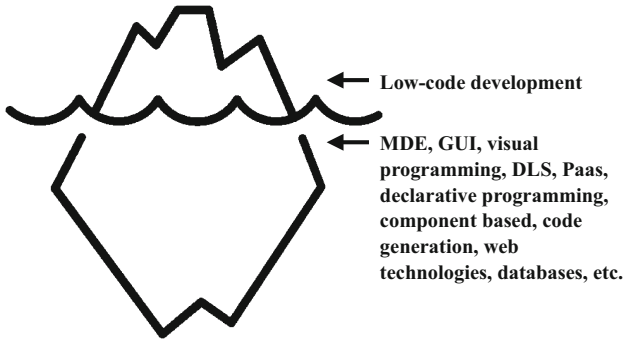


Fig. 17 Low-code iceberg

Consistently with this reasoning, the LCD platform provider Mendix[6] distinguishes four different kinds of user: citizen developer, professional developer, enterprise architect, and IT leader. In particular, the first two are supposed to use LCD tools to develop applications, while the latter two are responsible for selecting, adopting, configuring, and customising those LCD tools. Interestingly, the features and issues experienced by enterprise architects and IT leaders concerning their refinement and customisation tasks are not covered by the reviewed literature. In other words, most of the current literature focuses on the iceberg's tip in Fig. 17, while the submerged part is still not accounted for. Also the fact that almost half of the analysed peer-reviewed studies were published at workshops indicated that research on LCD is still in its maturation phase. In our opinion, this testifies the positioning of LCD technology adoption hype being in its *peak of inflated expectations* phase. As a consequence, the early stage in the maturation process of LCD pushes research and practice efforts to mostly target the solutions for citizen/professional developers rather than the platform support for LCD ecosystems.

From a historical perspective, LCD-based methods seem to follow a development path similar to MDE-based ones. Similarly to LCD, also MDE in its early days triggered huge interests thanks to ready-made solutions based on automated code generation. Subsequently, the wider adoption of MDE solutions moved the research focus from specific methods/tools to the methodology as a whole, especially due to customisation needs. In fact, major efforts were typically moved to tool providers that were commissioned the development of customised MDE solutions. As a consequence, research efforts have been devoted to the development of, among others, language engineering workbenches and model transformation languages to aid MDE experts in designing and implementing new solutions as well as refine and customise existing ones.

[6] https://mendix.com

In a very similar way, a dramatic growth in the adoption of LCD will necessarily raise questions related to portability, maintainability, and scalability both of the LCD tools themselves and the applications that can be generated via them, mention a few of desired properties. Indeed, several surveyed works hint as potentially relevant lines of research and development on LCD the lack of and need for specific LCD solutions relevant to business-critical contexts, notably testing and security. Moreover, already for current tools many adopters are concerned about LCD solutions and potential vendors lock-in, customisation, and adaptability [74]. As it also happened for MDE, it is foreseeable that upcoming research efforts on LCD will be dealing with the platforms themselves and specifically on how to support extension/customisation needs coming from citizen and professional developers. Hopefully, the past experience with MDE will serve as lessons learnt to avoid making the same mistakes related to tools usability, flexibility, and maintenance.

6 Threats to validity

When conducting this research, we used well-established guidelines for systematic studies, which includes the definition of a detailed research protocol that has been validated by external and independent researchers. Nevertheless, we acknowledge that validity threats may have affected our research. Hereafter, we discuss potential validity threats and related mitigation strategies.

6.1 Threats to external validity

External validity threats refer to the generalisation of causal findings concerning the desired population and settings [87]. One of the main threats to external validity affecting systematic reviews is that the selected papers may not be representative of the state-of-the-art and -practice of LCD. To mitigate such a potential threat, we targeted four different databases and indexing systems among the largest and most complete in the field of software engineering [22]. Besides, we complemented the automatic search with closed recursive backward and forward snowballing. Eventually, we complemented the search for scientific, peer-reviewed publications with a grey literature search. During the selection of primary studies, we excluded studies not written in English. While excluding studies written in other languages may affect the external validity of our research, we believe that such a threat is minimal as English is the *de-facto* standard language for scientific documents especially in computer science and software engineering.

Another possible external validity threat could be the existence of other terms used in place of LCD, that might have

limited the coverage of our search. In particular, it is not rare to find the term No Code as a nuance of LCD, used to distinguish those solutions for which no code writing is required at all. Nonetheless, our empirical observations demonstrate that in general No Code solutions are intended to be subsets of LCD, and indeed frequently LCD tools provide users with No Code features for selected problems. As a consequence, we can be confident that the LCD search also covered "no code only" approaches.

6.2 Threats to internal validity

Internal validity threats relate to poor settings impacting the design of the study [87]. We designed and conducted this research using well-established guidelines for systematic and multi-vocal studies. This helped us in minimising potential threats to internal validity. Concerning the validity of the analysed and synthesised data, we mitigated possible threats by using descriptive statistics. Besides, we cross-analysed the different categories of the extraction form and performed sanity checks on the extracted data. These tasks helped us identifying and solving potential issues on the consistency of the extracted data

6.3 Threats to construct validity

Construct validity threats affect the ability to derive a correct conclusion from the relations between treatment and outcome [87]. As already argued for threats to external validity, we performed the initial automated search using four different data sources, we complemented it with recursive snowballing activities and a grey literature search. A possible threat to construct validity may be caused by badly designed search strings. For the peer-reviewed search, we used a simple search string, so no particular attention needed to be paid for its construction. For the grey literature search, we used a more focused string that helped us in avoiding an overwhelming number of less relevant results. We screened the initial set of studies elicited from the automatic search according to well-documented selection criteria.

6.4 Threats to conclusion validity

Conclusion validity threats affect the relationships between the extracted data and obtained findings [87]. To mitigate these threats, we constantly and systematically applied and documented well-defined processes for systematic studies. Besides, we provided a complete and public replication package, which allows to reproduce each step of our study. All the authors participated in the definition of the extraction form, as well as in the data extraction, analysis and synthesis steps. Besides, we built the extraction form using well-established taxonomies and collecting values arising from the set of primary studies.

7 Related work

While LCD is already responsible for a fair share of industrial application development activities, its related research can still be considered to be at its initial stage. Researchers are trying to get hold of how LCD is growing and what needs shall be tackled to make it more effective for a broader audience. To the best of our knowledge, this work is the first multi-vocal systematic mapping study providing an analysis of core characteristics of LCD, notably founding paradigms, application domains, tools, and expected benefits. Nevertheless, there are several studies that shed some lights on specific aspects of LCD.

Sahay et al. propose a technical survey of eight relevant LCD platforms [72]. First, they elicit critical features provided by those platforms. Later, they compare the available alternatives based on the elicited characteristics. Consequently, their work mainly focuses on distinctive development facilities included in the platforms taken into account. In our study, we highlight founding paradigms, application domains, and benefits of LCD, not necessarily implemented in a concrete tool or platform. Besides, we provide a catalogue of LCD platform and tools used in research activities. However, we do not aim at comparing these platforms or tools.

Sanchis et al. provide another research focusing on eliciting relevant features that LCD platforms are supposed to provide [74]. However, the work by Sanchis et al. focuses on LCD for the manufacturing domain, only. Sanchis et al. use practitioners interviews for eliciting both LCD features and expected benefits. Besides, the authors include a more general discussion about reasons why (not) using LCD. Compared to our work, the work by Sanchis et al. also focus on features and benefits as being core characteristics of LCD. However, their work has a much narrower scope as it is limited to the manufacturing domain, only. Eventually, our work uses a different empirical research method rather than surveys.

In September 2020, the global research and advisory firm Gartner published the Gartner magic quadrant for LCD platforms [83]. The Gartner magic quadrant helps "*assessing how well technology providers are executing their stated visions and performing against Gartner market view*" [83]. The Gartner magic quadrant for LCD platforms evaluates 18 vendors. In our work, we elicit a catalogue of LCD platforms and tools, which includes the 18 tools reviewed from Gartner. However, as mentioned above, we do not provide an evaluation or a comparison of the elicited tools.

Ihirwe et al. analyse available platforms for developing IoT applications [44]. Interestingly, the authors include both MDE and LCD solutions and sketch a relation between them. In particular, they consider MDE as a mean for developing LCD applications. Based on the elicited data, we believe that LCD may be seen as a set of tools in a broader methodological context often being MDE. Moreover, we do not restrict our research to the IoT domain, only.

In their empirical study, Alamin et al. analyse thousands of posts containing discussions on 9 popular LCD platforms [12]. By analysing such discussions, they provide answers to three main research questions being: i) types of topics discussed, ii) distribution of topics across the LCD life cycle phases, and iii) perceived topics difficulty. Concerning the topics, Alamin et al. found 13 different topics, mostly focusing on development-related issues. According to their data, topics related to "Dynamic Event Handling" are the most difficult to answer. The research by Alamin et al. can be seen as complementary to this research and focusing more on practical issues related to LCD.

In his opinion paper [25], Cabot focuses on LCD-related questions such as whether or not LCD is a new method, whether and if it may relate to MDE, etc. Based on his experience in the MDE community, Cabot argues that he does not "*believe there is any fundamental technical contribution in low-code trend*". Despite his criticism, he sees LCD as an opportunity for bringing MDE into new research and industrial communities. In our study, we identified a rather strong relation both between LCD and MDE, but even between LCD and other technical paradigms such a visual programming, GUI-based development. Moreover, several core benefits of LCD seem to be in line with those of these paradigms, too.

8 Conclusion and future work

In this article, we reported on the planning, execution and results of a multi-vocal systematic review, which provides a snapshot of what LCD is considered to be both, especially concerning modelling and MDE. From an initial set of 720 peer-reviewed publications and 199 grey literature sources, we selected 58 primary studies. We analysed these studies using a precise data extraction, analysis, and synthesis process. The main highlights of our results are:

- While the term "low-code" was coined in 2014, first peer-reviewed publications on LCD appeared only in 2018. Since then, the publication trend on LCD-related topics has been growing considerably. Nonetheless, almost half of the analysed peer-reviewed studies were published at workshops, indicating LCD research being in a maturation phase.
- MDE is by far the most prominent core technology among the 27 considered in our primary studies as important for LCD. Correspondingly, improved return on investment, abstraction, graphic user interface-based development, and automation are the most mentioned benefits linked to the adoption of LCD.
- The grey literature tends to emphasise the rapid prototyping and usability features of LCD solutions, which are expected to disclose trial-and-error ways to modernisation without large upfront investments (as opposed to MDE and other software engineering methods).
- There is an extensive catalogue of 39 tools supporting LCD. In this respect, both peer-reviewed and grey literature focus on solutions for end-users, while the issues related to the creation of new/customised LCD platforms are largely neglected.

Based on our results, we believe that low-code development can be defined as a set of methods and/or tools in the context of a broader methodology, often being identified as model-driven engineering. Future work may encompass an additional review of the characteristics and complexity of applications built using low-code development, with the aim of assessing the state-of-the-practice of low-code development and reasoning about metrics such as scalability and performance.

Acknowledgements The work in this paper has been supported by the Swedish Knowledge Foundation (KKS) through the A-CPS, HERO, and SACSys projects and by the Sweden's Innovation Agency (VINNOVA) through the PANORAMA and BUMBLE projects.

Funding Open access funding provided by Mälardalen University.

References

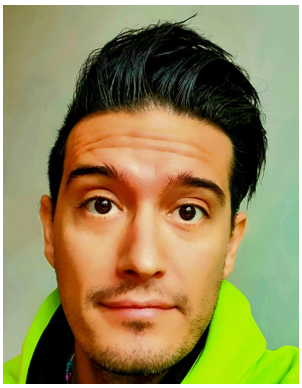
1. Futureproof your business with lasting agility. https://www.thaipr.net/en/business_en/3019857. Accessed: 2021-06-01
2. Genio. https://quidgest.com/en/about-quidgest/genio-platform/. Accessed: 2021-06-01
3. Low code / no code development platform. https://aimultiple.com/low-code-platform. Accessed: 2021-06-01

4. Low-code and ux: Can they work together? https://lichtenstein.io/low-code-ux. Accessed: 2021-06-01
5. Low-code. high impact. https://www.mendix.com/low-code-guide/low-code-use-cases/. Accessed: 2021-06-01
6. Low-code without low-code limitations. https://lansa.com/wp-content/uploads/2019/11/visual-lansa-in-detail-1.pdf. Accessed: 2021-06-01
7. Mendix versus outsystems - make an informed decision. https://marutitech.com/mendix-vs-outsystems/. Accessed: 2021-06-01
8. Siemens digital industries software. https://www.cimdata.com/en/education/plm-conferences/plmrm-pdt-spring-2021/sponsors/siemens. Accessed: 2021-06-01
9. The low-code guide. https://www.mendix.com/low-code-guide/. Accessed: 2021-06-01
10. What is it like to use j2paas low-code platform to develop software? https://programmersought.com/article/72885313673/. Accessed: 2021-06-01
11. *Procs of the 1st Intl Workshop on Model-driven Engineering By Example co-located with MODELS 2013, Miami, Florida, USA, September 29*. CEUR Workshop Proceedings, (2013)
12. Al Alamin, M.A., Malakar, S., Uddin, G., Afroz, S., Haider, T.B., Iqbal, A.: An empirical study of developer discussions on low-code software development challenges. *arXiv e-prints*, p. 2103, (2021)
13. Benjamin, A., Sven, H., Alexander, N.: App development via low-code programming as part of modern industrial engineering education. In: International Conference on Applied Human Factors and Ergonomics, pp. 45–51. Springer, Berlin (2020)
14. Adrian, O.: Introducing model-driven apps – a new way to create. https://powerapps.microsoft.com/sv-se/blog/introducing-model-driven-apps/, (2020). Accessed: 2021-06-01
15. Adrian, P.: What is low-code? https://www.alibabacloud.com/blog/what-is-low-code_597659, (2021). Accessed: 2021-06-01
16. Ali, N.B., Petersen, K.: Evaluating strategies for study selection in systematic literature studies. ACM, In Proceedings of ESEM (2014)
17. Qurat ul ain, A., Dimitris, K., Konstantinos, B.: Efficiently querying large-scale heterogeneous models. In *Proceedings of MODELS-C*, pp. 1–5, (2020)
18. Almonte, L., Cantador, I., Guerra, E., de Lara, J.: Towards automating the construction of recommender systems for low-code development platforms. In *Proceedings of MODELS-C*, pp. 1–10 (2020)
19. Eager, A.: Enterprise software and application services. extract - digital enablement via low code platforms: understanding the position, uncovering the prospects. https://info.unit4.com/rs/900-SZD-631/images/U4-ALL-GEN-AR-Digital-Enablement-via-Low-Code-Platforms.pdf, (2019). Accessed: 2021-06-01
20. Arora, R., Ghosh, N., Mondal, T.: Sagitec software studio (s3)-a low code application development platform. In *Proceedings of I4Tech*, pp. 13–17. IEEE, (2020)
21. Basili, V.R., Caldiera, G., Dieter, R.H.: The goal question metric approach. Encycl. Softw. Eng. **2**, 528–532 (1994)
22. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. **80**(4), 571–583 (2007)
23. Brunschwig, L., Guerra, E., de Lara, J.: Towards access control for collaborative modelling apps. In *Proceedings of MODELS-C*, pp. 1–10 (2020)
24. Burnett, M.M., McIntyre, D.W.: Visual programming. COMPUTER-LOS ALAMITOS **28**, 14–14 (1995)
25. Cabot, J.: 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*, pp. 1–3 (2020)
26. Case, A.F.: Computer-aided software engineering (case) technology for improving software development productivity. ACM SIGMIS Datab. **17**(1), 35–43 (1985)
27. Charmaz, K., Belgrave, L.L.: Grounded theory. The Blackwell encyclopedia of sociology (2007)
28. Garms, C.: Get the right mix of low-code and pro code to modernize your enterprise applications. https://diginomica.com/right-mix-low-code-pro-code-enterprise-applications, (2020). Accessed: 2021-06-01
29. Vanner, C.: What is low-code automation and how could it benefit you? https://www.bizagi.com/en/blog/agile-low-code/what-is-low-code-automation-how-could-it-benefit-you, (2021). Accessed: 2021-06-01
30. Colantoni, A., Berardinelli, L., Wimmer, M.: Devopsml: towards modeling devops processes and platforms. In *Proceedings of MODELS-C*, pp. 1–10, (2020)
31. Cruzes, D.S., Dyba, T.: Recommended steps for thematic synthesis in software engineering. In *Proceedings of ESEM*, pp. 275–284. IEEE (2011)
32. Di Sipio, C., Di Ruscio, D., Nguyen, P.T.: Democratizing the development of recommender systems by means of low-code platforms. In *Proceedings of MODELS-C*, pp. 1–9 (2020)
33. Dimanidis, A., Chatzidimitriou, K.C., Symeonidis, A.L.: A natural language driven approach for automated web api development: Gherkin2oas. In *Companion Proceedings of the The Web Conference 2018*, pp. 1869–1874 (2018)
34. Erlend Barstad Strand. Evaluating low-code platforms: What's under the hood? https://www.genus.no/blog/evaluating-low-code-platforms/. Accessed: 2021-06-01
35. Estefan, J.A.: Survey of model-based systems engineering (MBSE) methodologies, Rev. B. Technical Report INCOSE- TD- 2007-003- 01, International Council on Systems Engineering, Seattle, WA, May 23, (2008). MBSE Initiative
36. Brown, E.: Model driven development versus code generation. what's the right balance? https://www.itbriefcase.net/model-driven-development-vs-code-generation, (2020). Accessed: 2021-06-01
37. Franzosi, R.: *Quantitative narrative analysis*. No. 162. Sage, (2010)
38. Garousi, V., Felderer, M., Mäntylä, M.V.: Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf. Softw. Technol. **106**, 101–121 (2019)
39. Mironescu, G.: Wem: No code development to democratize cloud-based application functionality creation. https://wem.io/wp-content/uploads/2019/06/IDC-Technology-Spotlight-on-WEM.pdf. Accessed: 2021-06-01 (2018)
40. Greenhalgh, T., Peacock, R.: Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. BMJ **331**(7524), 1064–1065 (2005)
41. van der Weel, H.: Low-code/no-code in an enterprise landscape. https://www.syntouch.nl/low-code-no-code-in-an-enterprise-landscape/, (2017). Accessed: 2021-06-01
42. Henriques, H., Lourenço, H., Amaral, V., Goulão, M.: Improving the developer experience with a low-code process modelling language. In *Proceedingss of MODELS*, pp. 200–210 (2018)
43. Horváth, B., Horváth, Á., Wimmer, M.: Towards the next generation of reactive model transformations on low-code platforms: three research lines. In *Proceedings of MODELS-C*, pp. 1–10, (2020)
44. Ihirwe, F., Di Ruscio, D., Mazzini, S., Pierini, P., Pierantonio, A.: Low-code engineering for internet of things: A state of research. In *Procs of MODELS-C*, pp. 1–8 (2020)
45. van Kooten, J.: Do model-driven development (mdd) platforms make good on their promises? https://www.bpmcompany.eu/en/do-model-driven-development-mdd-platforms-make-good-on-their-promises/, (2016). Accessed: 2021-06-01

46. den Haan, J.: Low-code principle #1: Model-driven development, the most important concept in low-code. https://www.mendix.com/blog/low-code-principle-1-model-driven-development/, (2020). Accessed: 2021-06-01
47. den Haan, J.: Introducing the low-code manifesto. https://www.mendix.com/blog/introducing-the-low-code-manifesto/, (2021). Accessed: 2021-06-01
48. den Haan, J.: Process meets intelligence: from workflow to automated enterprise. https://www.forbes.com/sites/forbestechcouncil/2021/05/27/process-meets-intelligence-from-workflow-to-automated-enterprise/?sh=2c06139f232d, (2021) Accessed: 2021-06-01
49. Niiranen, J.: What will power fx mean for model-driven power apps? https://jukkaniiranen.com/2021/03/what-will-power-fx-mean-for-model-driven-power-apps/, (2021). Accessed: 2021-06-01
50. Khorram, F., Mottu, J.-M., Sunyé, G.: Challenges and opportunities in low-code testing. In *Proceedings of MODELS-C*, pp. 1–10 (2020)
51. Kitchenham, B., Brereton, P.: A systematic review of systematic review process research in software engineering. Inf. Softw. Technol. **55**(12), 2049–2075 (2013)
52. Kitchenham, B.A., Charters, S..: Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Keele University and University of Durham, (2007)
53. Kolovos, D.S., García-Domínguez, A., Rose, L.M., Paige, R.F.: Eugenia: towards disciplined and automated development of GMF-based graphical model editors. Softw. Syst. Model. **16**(1), 229–255 (2017)
54. Kourouklidis, P., Kolovos, D., Matragkas, N., Noppen, J.: Towards a low-code solution for monitoring machine learning model performance. In *Procs of MODELS-C*, pp. 1–8, (2020)
55. Liebel, G., Marko, N., Tichy, M., Leitner, A., Hansson, J.: Assessing the state-of-practice of model-based engineering in the embedded systems domain. In *International Conference on Model Driven Engineering Languages and Systems*, pp. 166–182. Springer, Berlin (2014)
56. Burnham, M.: Low-code support. https://www.timeseries.com/top-evaluation-criteria-for-selecting-a-low-code-platform-part-2/, (2021). Accessed: 2021-06-01
57. Burnham, M.: Top evaluation criteria for selecting a low-code platform - part 2. https://www.timeseries.com/top-evaluation-criteria-for-selecting-a-low-code-platform-part-2/, (2021). Accessed: 2021-06-01
58. Malavolta, I., Lago, P., Muccini, H., Pelliccione, P., Tang, A.: What industry needs from architectural languages: a survey. IEEE Trans. Softw. Eng. **39**(6), 869–891 (2012)
59. Martins, R., Caldeira, F., Sá, F., Abbasi, M., Martins, P.: An overview on how to develop a low-code application using outsystems. In *Proceedings of ICSTCEE*, pp. 395–401. IEEE (2020)
60. Mendix. Low code use cases. https://www.mendix.com/low-code-guide/low-code-use-cases/. Accessed: 2021-04-12
61. Metrôlho, J., Araújo, R., Ribeiro, F., Castela, N.: An approach using a low-code platform for retraining professionals to ict. In *Proceedings of EDULEARN*, pp. 7200–7207. IATED, (2019)
62. Metrôlho, J., Ribeiro, F., Araújo, R.: A strategy for facing new employability trends using a low-code development platform. In: *14th International Technology, Education and Development Conference*, pp. 8601–8606, (2020)
63. Pantelimon, S.-G., Rogojanu, T., Braileanu, A., Stanciu, V.-D., Dobre, C.: Towards a seamless integration of iot devices with iot platforms using a low-code approach. In *Proceedings of WF-IoT*, pp. 566–571. IEEE, (2019)
64. Peinl, R., Perak, O.: Bpmn and dmn for easy customizing of manufacturing execution systems. In *International Conference on Business Process Management*, pp. 441–452. Springer, Berlin (2019)
65. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In *Proceedings of EASE*, pp. 68–77 (2008)
66. Philippe, J., Coullon, H., Tisi, M., Sunyé, G.: Towards transparent combination of model management execution strategies for low-code development platforms. In Proceedings of MODELS-C, pp. 1–10 (2020)
67. Richardson, C., Rymer, J.R., Mines, C., Cullen, A., Whittaker, D.: New development platforms emerge for customer-facing applications. Forrester, Cambridge (2014)
68. Spiegel, R.: Siemens plm becomes siemens digital industries software. https://www.designnews.com/automation-motion-control/siemens-plm-becomes-siemens-digital-industries-software. Accessed: 2021-06-01
69. Rodgers, M., Sowden, A., Petticrew, M., Arai, L., Roberts, H., Britten, N., Popay, J.: Testing methodological guidance on the conduct of narrative synthesis in systematic reviews: effectiveness of interventions to promote smoke alarm ownership and function. Evaluation **15**(1), 49–73 (2009)
70. Rossi, G., Urbieta, M., Distante, D., Rivero, J.M., Firmenich, S.: Years of model-driven web engineering. What we achieved, What is missing. CLEI Electr. J. **19**(5–57), 12 (2016)
71. Black, R.: 5 faqs on the low-code approach. https://searchsoftwarequality.techtarget.com/feature/5-FAQs-on-the-low-code-approach, (2020). Accessed: 2021-06-01
72. Sahay, A., Indamutsa, A., Di Ruscio, D., Pierantonio, A.: Supporting the understanding and comparison of low-code development platforms. In *Proceedings of SEAA*, pp. 171–178. IEEE, (2020)
73. Sakhnyuk, P.A., Sakhnyuk, T.I.: Intellectual technologies in digital transformation. In *IOP Conference Series: Materials Science and Engineering*, vol. 873, p. 012016. IOP Publishing, (2020)
74. Sanchis, R., García-Perales, Ó., Fraile, F., Poler, R.: Low-code as enabler of digital transformation in manufacturing industry. Appl. Sci. **10**(1), 12 (2020)
75. Schmidt, D.C.: Model-driven engineering. Computer-IEEE Comput. Soc. **39**(2), 25 (2006)
76. Sendall, S., Kozaczynski, W.: Model transformation: the heart and soul of model-driven software development. IEEE Softw. **20**(5), 42–45 (2003)
77. Khoshafian, S.: No-code/low-code: Why you should be paying attention. https://venturebeat.com/2021/02/14/no-code-low-code-why-you-should-be-paying-attention/, (2021). Accessed: 2021-06-01
78. Shaw, M.: What makes good research in software engineering? Proceedings of STTT **4**(1), 1–7 (2002)
79. Silva, C., Vieira, J., Campos, J.C., Couto, R., Ribeiro, A.N.: Development and validation of a descriptive cognitive model for predicting usability issues in a low-code development platform. *Hum. Factors* (2020)
80. Kovacevic, S.: Low code applications and model-driven development. https://www.linkedin.com/pulse/low-code-applications-model-driven-development-srdjan-kovacevic-phd/, (2020). Accessed: 2021-06-01
81. Tisi, M., Mottu, J.-M., Kolovos, D., De Lara, J., Guerra, E., Di Ruscio, D., Pierantonio, A., Wimmer, M.: Lowcomote: Training the next generation of experts in scalable low-code engineering platforms. In *Proceedings of MDE@DeRun* (2019)
82. VerkstadsForum. Low-code platforms - hot solutions expected to be doubled within two years: Mendix visionary market leader according to gartner. https://plm-erpnews.se/low-code-platforms-hot-solutions-expected-to-be-doubled-within-two-years-mendix-visionary-market-leader-according-to-gartner/, (2021) Accessed: 2021-06-01

83. Vincent, P., Natis, Y., Iijima, K., Wong, J., Ray, S., Jain, A., Leow, A.: Critical capabilities for enterprise low-code application platforms. https://www.gartner.com/en/documents/3991223. Accessed: 2021-04-12
84. Waszkowski, R.: Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine **52**(10), 376–381 (2019)
85. Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., Heldal, R.: A taxonomy of tool-related issues affecting the adoption of model-driven engineering. Softw. Syst. Model. **16**(2), 313–331 (2017)
86. Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Require. Eng. **11**(1), 102–107 (2006)
87. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Computer Science, Springer, Berlin (2012)
88. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of EASE*, pp. 38:1–38:10. ACM, (2014)
89. Yuanling, Z., Plus, Y.: Alibaba's solution for low-code development. https://www.alibabacloud.com/blog/yida-plus-alibabas-solution-for-low-code-development_595486?spm=a2c65.11461447.0.0.7462759fDy3ECU, (2019). Accessed: 2021-06-01
90. Zolotas, C., Chatzidimitriou, K.C., Symeonidis, A.L.: Restsec: a low-code platform for generating secure by design enterprise services. Enterp. Inf. Syst. **12**(8–9), 1007–1033 (2018)

Alessio Bucaioni is a software engineer currently working as assistant professor in computer science at Mälardalen University. He received his Ph.D degree from Mälardalen University in 2018. His research focuses on several aspects of the development of complex software-intensive systems from software architecture to model-driven development.



Antonio Cicchetti is an Associate Professor at Mälardalen University (Sweden). His research interests include modelling languages engineering and evolution, model transformations, multi-paradigm modelling, model versioning, combination of MDE and CBSE for complex systems, industrial software engineering, and software engineering (for) gamification. More info at: http://www.es.mdh.se/staff/198-Antonio_Cicchetti

Federico Ciccozzi is an Associate Professor at Mälardalen University (Sweden). His research specializes in: definition of DSMLs, model transformations, system properties preservation, multi-paradigm modelling, model versioning, combination of MDE and CBSE for complex systems, blended modelling, language and compiler engineering. More info at: http://www.es.mdh.se/staff/266-Federico_Ciccozzi