DevOps Practitioners' Perceptions of the Low-code Trend

Saima Rafi
Department of Informatics and Systems
University of Murcia
Spain
saeem112@gmail.com

Mary Sánchez-Gordón
Department of Computer Science
Østfold University College
Halden Norway
mary.sanchez-gordon@hiof.no

Muhammad Azeem Akbar
Department of Software Engineering
LUT University
Finland
azeem.akbar@lut.fi

Ricardo Colomo-Palacios
Department of Computer Science
Østfold University College
Halden Norway
ricardo.colomo-palacios@hiof.no

ABSTRACT

Background: DevOps is currently one of the main trends in software development. Low-Code is also an emerging tendency that, combined with DevOps, may offer significant value to software businesses by improving the process. However, how DevOps practices and low-code are combined is little known. **Aim:** This study aims to understand the practitioner's perspectives on low-code trends. **Method:** Twelve interviews with IT professionals who deal with low-code in the context of DevOps were conducted. Then, a grounded theory approach was used to theme the interview quotes into emergent categories. **Results:** The main result of this exploratory study reveals that such an approach is the most common response to the skill shortages of software professionals. **Conclusion:** This study suggests the emergence of DevOps and low-code could significantly contribute to the development of quality products with low-cost and time.

CCS CONCEPTS

• Software and its engineering → Software creation and management

KEYWORDS

DevOps, Low-code, Interviews, Grounded theory, SPI

1 Introduction

Nowadays, in a fast changing environments, DevOps is one of the most effective approaches to shorter response times to business needs [1]. In DevOps, the collaboration of development and operations teams is increasing the flexibility of the software development process. However, to perform continuous development, continuous testing, continuous deployment, and continuous delivery; agility is needed in all phases of a DevOps lifecycle [2]. In turn, low-code approaches seem, initially, great tools to extend the gains of agility in software organizations with faster response and low operational costs [3]. Low-code is seen as a visual and semi-automated set of tools to construct software [4]. It delivers software products with minimum effort to write code using a graphical user interface (GUI) and requires the least possible effort for the installation, deployment, and configuration of software-based solutions. Low-code solutions could be a promising step to meet the rapid demands of software development [5].

The stream of low-code software development has grown due to the incorporation of low-code solutions to large software vendors product portfolios [6]. According to a Forrester report [7], the low-code platform market is expected to be $21 Billon by 2022 while a Gartner report [8] states that around 65% of large companies will use low-code platforms to some extent by 2024. Low Code Development Industry is projected to achieve a global market size of US$ 187 Bn by 2032 [9]. However, there are many concerns related to the use of low-code tools to build applications instead of traditional approaches, [5].

Observing that both low-code and DevOps approaches share a common motive of delivering valuable software products; we formulate the following research question for this study: What do practitioners think about the emergence of DevOps and low-code? The collaboration of automated tool kits to accelerate a manual process of development and culture change could bridge the gap between development and operations, to take the business wave to next level [10]. To explore the usefulness of low-code software development platforms inside the DevOps paradigm, we have conducted interviews with twelve IT professionals. Moreover, the exploratory nature of this study calls for a qualitative research approach like grounded theory (GT) [11].

The paper is organized as follows. Section 2 presents the background and related work. Section 3 defines the research methodology while Section 4 presents the results. Then, Section 5 discusses the results, and Section 6 draws some conclusions.

2 Background and related works

DevOps plays a major role in building a culture in which development and operations teams work in a collaborative environment [12]. DevOps is one step forward to agile as it encourages continuous development and delivery cycles. DevOps had its origin at Agile Conference in 2008 [13] and currently, it is a worldwide phenomenon in all kinds of organizations [14]. DevOps has been defined in various contexts by different researchers. For example, Dyck et al. [15] DevOps is "a cross-functional collaboration between teams". According to Erich et al. [16] "to support development and operation teams the DevOps is a conceptual framework". The overall objective of DevOps is to increase customer satisfaction by providing continuous services and updates.

The idea of low-code development is not new, however, the new generation of tools based on a Platform-as-a-Service (PaaS) model, e.g., (Google) App Maker and (Microsoft) PowerApps, are reaching leading positions in specific domains such as mobile applications and database applications [20]. Low-code development approaches are based on graphical user interfaces (GUI) comprising a set of templates, wizards, and drag and drop options that help to build the software product rapidly with minimal effort of self-coding [17]. A Forrester report [7] defined low-code as a "platform available to customers for software development with visual and declarative techniques instead of hand-code at low cost".

Van der Burgh [18] gave an initial idea towards the adoption of low-code development in DevOps (LCDevOps-RSA) by characterizing low-code as a tool-based approach to design business software applications and by establishing a readiness model that shows the relationships between both approaches. Philippe et al. [19] conducted a study that highlighted the complexity of code development and elaborate on the importance of low-code to minimize code development complexity. Therefore, such a mixed approach could bring a significant benefit to developing and delivering high-quality software with minimum time and cost. However, Tisi et al. [20] highlight three main limitations: (i) Scalability to build large-scale and mission-critical enterprise applications, (ii) Fragmentation due to each provider proposing its own low-code development paradigm, which is linked to a certain programming model, and (iii) Software-only systems build by potential users with little knowledge of programming but they could be experts (citizen developers) in some other domain and expect to use their knowledge in the application, at the appropriate level of abstraction and using familiar formalisms.

In this exploratory study, we aim to understand the DevOps practitioners' perceptions of the low-code trend by analizing emerging categories in the light of software process improvement (SPI). Given that the state-of-the-art knowledge on SPI is compiled in the SPI manifesto [21], it was used to map the findings and provide an overview of the topic. The SPI manifesto measures the social, human, and organizational aspects to improve the software development process in a productive manner [21]. For example, Khan and Shameem [22] used it to create a taxonomy of the key factors that could impact the adaptation and implementation of DevOps practices.

3 Research Methodology

GT is useful to develop a substantive concept for qualitative research where we have questions like "what is going on in this area?" [23]. Although this approach is widely used in social studies, it is also used by software engineering researchers [24], [25]. GT presents different approaches. Glaser focuses on the "emergence (of research questions, of codes, of theory)" [26], while Strauss and Corbin focus on a "systematic approach and validation of criteria" [23]. In addition, Charmaz [27] emphasizes the "role of researchers on theory".

In this study, we used a classic GT i.e., Glaser's approach, which keeps researchers' attention focused on the data and requires that any concept be grounded in the data [28]. We cannot conduct a case study or a survey for this study as there is no clear hypothesis in front [29] and DevOps is a social-technical process in which a case study that is based on one perspective provides only limited insights into concepts and human behaviors [30].

3.1 Data collection

We conducted interviews with 12 experts to get their opinions on merging DevOps with low-code with the aim to accelerate innovation and continuous software delivery. The process of data collection should stop when a saturation point is reached, i.e., no new concept or idea is being obtained [11], but we do not claim a saturation point in this exploratory study, yet. After finding the concepts and categories that represent the main concept of the proposed research objective, the opinions of participants were collected. This step clarifies the researchers to decide what data should be collected in the next steps.

To recruit participants, an invitation was sent to twenty-five IT professionals from three different countries, who are working with DevOps and low-code. They were interviewed using Skype, Zoom, and Teams (based on participants' choice). We used open-ended questions to clarify the concept overall. We asked their opinion about merging DevOps with low-code approaches to accelerate innovation and continuous software delivery, their suggested strategies, and their opinions on handling risks. The next set of questions was about business values and roles after adopting DevOps and low-code in their software development environment. For example, 1) What are the values of software process improvement for your business projects? 2) How does DevOps help you in managing automation and team tasks? The interview sessions lasted around 40 minutes and they were recorded and converted to transcripts for later analysis.

3.2 Data analysis

Data analysis is based on "data coding" which is an important step of GT [11]. It consists of two processes: substantive, and theoretical coding. Glaser [26] defines them as "Substantive codes are the emergent categories and properties that conceptually describe the phenomenon under study, whereas the theoretical codes are the emergent abstractions that model the integration of substantive codes as an interrelated set of hypotheses for resolving the main concern". The steps, how GT was employed to code the collected data is available on link https://tinyurl.com/28sp2hne.

A mapping of categories and codes with SPI manifesto was also conducted to describe the impact of integrating DevOps and low-code development in terms of (i) people: to analyze how the integration of DevOps and low-code affects the daily activities of stakeholders such as end-users, developers, and IT team members; (ii) business: to measure benefits of using low-code and DevOps to make businesses successful; and (iii) change: to manage organizational change and focus on understanding the concept of improvement in the software development process in the current situation, i.e., economic crisis and social impact on businesses due to COVID-19.

4 Results

In this section, the participants' demographics; and the opinions of practitioners concerning the combination of DevOps and low-code are discussed.

4.1 Demographics

Table 1 shows the demographic information. The first column "Id" shows an identifier to anonymize each participant. Participants reported various roles: IT manager (ITm), Project manager (PM), System analyst (SA), Operational manager (OPm), and Developer (Dev). They also reported more than four years of experience (Exp) and most of them came from Spain (5), followed by Finland (4), and China (3).

d	Role	Exp.	Domain	Tools	Country
	ITm	5	Telecom	Github, App maker	Finland
	Dev	> 4	Telecom	Github, App maker	Spain
	PM	+ 7	Software	Puppet, App maker	Spain
	OPm	5	Software	Puppet, App maker	China
	PM	10	Software	Chief, App maker, Appian	Finland
	OPm	> 5	Software	Puppet, PowerApps	China
	SA	+ 6	Banking	Jenkins	Finland
	SA	7	Banking	Jenkins, Gradle	China
	SA	5	Software	Chief, Puppet, PowerApps	Spain
0	ITm	4	Software	Puppet, PowerApps	Spain
1	Dev	+ 6	Telecom	Jenkins, TestOps	Finland
2	PM	+ 10	Telecom	Jenkins	Spain

Table 1: Demographic Information

4.2 SPI overview of low-code emergence in DevOps

The results of this study are explained by illustrating the emerged categories and codes in the light of SPI manifesto. SPI manifesto could be used with the intent to improve software development towards increased quality and productivity levels in a formal way [22]. The values of the SPI manifesto are people, change, and business [21].

Id	**Categories/** *sub-categories/* codes	Participants
P	People	31
P1	Reduce skill shortages of professionals	24
P1.1	Reduce continuous development and delivery	PM3, SA9
P1.2	No long-handwritten codes	PM4, OPm6, SA7, Dev11
P1.3	Reduce work of developers/ save energy of developers	ITm10, Dev2, OPm4, PM3, SA8
P1.4	Effective for citizen developers/non-professionals	SA8, Dev11, PM12
P1.5	User-friendly frameworks	SA7, SA9, ITm10, PM12
P1.6	No need for high-quality developers	PM3, SA8
P1.7	On-demand delivery	ITm1, OPm4, Dev11
P1.8	Can use developer's opinions in other tasks instead of coding	ITm1
P2	Speed up software development to meet customer demands	
P2.1	Simple code with GUI interface	OPm4, SA7
P2.2	Easy to modify	PM3
P2.3	No complex process which speeds up development	Dev2
P2.4	Deliver software products on time	OPm6, SA9
P2.5	Give priority to customers	SA8
B	Business	29
B1	Economic benefits	11
B1.1	Package of tools available	Dev2, ITm10, SA7
B1.2	Reduce financial issues/ within budget	OPm4, PM5, PM12
B1.3	Rapid development and delivery	ITm10, Dev2, OPm4, PM12
B1.4	Reduce IT cost	OPm6
B2	Performance monitoring	
B2.1	Help DevOps teams to focus on the enterprise's performance/ no need to worry about code	Dev2, ITm10, SA8
B2.2	Low-code can transform hours spent on repetitive tasks by a team, to focus on innovation	ITm1, PM3
B2.3	Automated dashboard for quality check	PM5
B2.4	Effective software delivery	OPm6
B2.5	Increase productivity and efficiency	OPm4
B3	Compliance	
B3.1	Help in updating and adding new functionalities	Dev11
B3.2	Must follow government regulations during application development	ITm1, PM12
B3.3	Reduce manual recheck of iterations	SA9
B3.4	Automated compliance frameworks	SA9
B3.5	Improve team autonomy to make decisions	ITm10
B4	Security and governance	
B	Reduce the use of unauthorized tools	PM3

4.1			
B 4.2	Use of monitoring tools to identify vulnerabilities		Dev2
B 4.3	Reliable product		OPm4, SA8
C	Change	15	
C 1	Infrastructure independence	4	
C 1.1	Supporting work-from-home		SA8, SA9
C 1.2	Low-code platforms help DevOps needs of streamlined configuration and management tools		Dev11
C 1.3	Building collaboration culture within teams		ITm1
C 2	Resilience and easy migration		
C 2.1	Bring resilience to an organization		PM5, PM12
C 2.2	Manage automation environment		SA7
C 2.3	Speed up response time		PM5, Dev2, ITm10
C 3	Consistency within an organization		
C 3.1	Low-code tools can integrate easily with automated platforms		PM5
C 3.2	Reduce complexity issues		Dev2
C 3.3	Do not affect the organization's workflow		OPm4
C 3.4	Effective management of project/application		SA8, PM12

#: number of times mentioned; Participant: role+Id from Table 1 (e.g., ITm1, PM3)

Table 2: Codebook

Table 2 shows the final version of the codebook. It also shows the mapping of all the emergence categories, subcategories, and codes with the SPI manifesto. The first two columns are self-explained. The third column "#" shows the number of times that a code appears in interview data (frequency) while the fourth column "participants" include the respective list of participants (role+Id from Table 1, e.g. Pm3). For instance, in the category "people" (Cp), Cp1.1 "reduce continuous development and delivery", was pointed out by two participants (PM3, SA9), similarly, Cp1.2 "no long handwritten codes" shows that four participants (PM4, OPm6, SA7, Dev11), mentioned this code during the interview session. It is noteworthy that the main category Cp1 "Reduce skill shortages of professionals" is the most commonly mentioned by the participants (24). Therefore, it is clear from the findings that low-code approaches combined with DevOps can help in reducing the need for human resources. It means the growing talent demands and available pool of software professionals can be fulfilled by enterprises by adopting low-code approaches. Thus, use low-code approaches can be used to develop software products alleviating the pressure to hire professional coders.

In what follows, we briefly describe each sub-categories. However, due to limited space, we present only the most significant quotes related to each sub-categories.

4.2.1 People

a) Reduce skills shortages of professionals. The use of low-code platforms in DevOps will positively impact an organization to produce valuable products. This is not limited to one part of people, roles, or activities. The use of low-code in DevOps environments will help developers to build and modify software products without writing complex codes. Low-code platforms provide user interfaces (UI) and support high-level programming abstractions, then, developers only must focus on logic and flow. This will help to reduce the amount of software personnel.

> "the current situation is leading DevOps teams to search for more developers as more enterprises are shifting their data on cloud. The low-code provides graphical user interface (GUI) and simple frameworks to code….the developers just have to cope-up the desired operations using these platforms without thinking about code…there is no need of recruiting new professional developers as we are all facing economic crisis" [Dev2].

b) Speed up software development to meet customer demands. Low-code promise to build a software application with simple code that is easy to modify speeds up the complete development process within an organization and reduces time and cost.

> "… using low-code platforms is not just a story for citizen developers to help them in building an application. From my perspective, both professional developers and citizen developers play an important role in building a software product that is not complex and speed up development and deployment tasks to meet customer needs and use of them in DevOps process will help to maintain development speed and cost" [OPm6].

4.2.2 Business

a) Economic benefits. Software organizations are facing problems like economic crises due to the current situation of COVID-19. These organizations are not interested in recruiting new developers and operational managers; they want all tasks to be performed on time by the current team. Similarly, many software development organizations are transferring their data to the cloud and are using DevOps as a cultural shift to accelerate the overall lifecycle. Considering low-code platforms can help organizations to save cost and time. It also suggests that there is no need of recruiting new team members as low-code tools will automate and speed up the entire software development process.

> "low-code is an economic friendly package of tools available in market….more than 200 low-code platforms are available…as they provide solutions to people who want to develop software projects….the DevOps environment needs all tasks to be performed in a continuous manner for which more team members are required causing financial issues and sometimes developers are not available … integrating low-code tools will help DevOps teams to maintain DevOps culture with rapid development and delivery of software

> product, eliminating the need to hire new developers" [ITm10].

b) Performance monitoring. In an organization, daily monitoring and log checks are essential factors to maintain business operations. In DevOps, these factors act as a backbone to ensure that complex interdependencies are running smoothly. Low-code dashboards can monitor all activities and provide an opportunity for the operations team to focus on innovation.

> "automated monitoring tools are required for DevOps … low-code platforms can transform hours spent on these repetitive tasks into opportunities for engineering teams to focus on innovation, while still ensuring the health and performance of enterprise software portfolios" [SA8].

c) Compliance. Regulations affect all economic sectors. Software practitioners must keep track of every regulatory requirement while updating or adding new functionalities to software. The integration of low-code tools to perform certain tasks will help in providing solutions for system improvements, but it needs certain regulations to fulfill regulatory requirements.

> "the continuous deployment is all what customer demands for … data privacy laws differ across nations and borders. Imagine having to manually recheck every iteration of your application deployments for compliance. Luckily, low-code operations mean having an automated compliance frameworks" [ITm1].

d) Security and governance. Software development organizations are always finding solutions to resolve security issues e.g., cyberattacks, authentication, unauthorized tools, and other security problems that can slow down the development process. The integration of low-code tools in DevOps may cause security and governance issues. Therefore, to configure security issues we need monitoring tools to identify vulnerabilities for quick feedback.

> "Platforms that use hybrid approach are useful but at the same time cause security problems … low-code approach in DevOps is a double-edge sword. They facilitate the development team but have issues like data security and governance … providing fast delivery to customers' shows that security is not a priority concern of citizen developers. If you are working for large organization security and governance even become more important…we are in early stage to say more about low-code approaches in DevOps but, using mature and reliable monitoring tools we can mitigate this problem" [Dev2]

4.2.3 Change

a) Infrastructure independence. Digital organizations are widely spreading and adopting cloud environments to mitigate their needs by allowing employees to either work from home (due to pandemic situations) or across multiple subunits. Cloud infrastructures make operational tasks more flexible for software organizations. In a multi-cloud environment, DevOps needs configuration and management tools to measure relationships between resources and keep track of compliance issues. Low-code can help in this context by providing tools and building collaboration among teams.

> "Enterprises are bypassing single-vendor lock-in to spread workloads across multiple cloud partners… that means DevOps needs streamlined configuration and management tools that can track relationships…for that use of low-code development tools can build a relationship that provides observability across the range of vendors consuming enterprise data" [SA8].

b) Resilience and easy migration. The adoption of DevOps and low-code can increase an organization's resilience to managing an automated development environment. The use of low-code in DevOps can speed up the overall response time.

> "Low-code platforms are great tools to extend the gains of Agile and DevOps approaches … Companies benefit from continuous delivery that ultimately fosters the productive collaboration between the business and IT" [SA 7].

c) Consistency within an organization. Low-code tools can integrate easily with automated deployment platforms. DevOps and low-code can increase the speed of development, and, at the same time, reduce complexity issues. This combination can bring a positive impact on software process.

> "Having one platform that can provide all this with 'no' or 'low' code can increase the speed of DevOps teams … the developers don't have to worry about the compatibility part…. the low-code platform easily gets integrated with the organization's workflow. There isn't any risk of disruption involved and make the teams confident" [PM12].

5 Discussion

Based on the collected evidence and the relationship of categories and themes with the SPI manifesto, we have presented the results of the combination of DevOps and low-code approaches. The classification of categories with SPI manifesto i.e., people, business, and change, reveals that there is an emerging theory about the DevOps paradigm with low-code approaches that will enhance business value by improving software processes, and the results of combining them will help in speeding development and deployment. From the findings, it is clear that DevOps and low-code share a common motivation. The discussion about how low-code platforms are useful for DevOps is as follows:

Bridging the gap between skill shortages and the available professional pool. Low-code helps novice or aspiring coders to build code without having to write complex lines of code. Using low-code approaches, highly skilled professionals can focus on core project areas and can spend time on other productive project activities. This means that by merging low-code approaches with DevOps, an organization can have a part of its workforce focusing on development which is now relatively faster and easy while the other part is focusing on other vital aspects e.g., scheduling meetings with operations team to discuss about performance structure of an application, improving collaborations.

Expediting application development. Software applications developed using low-code platforms are easily accessible and are not very complex [31]. Low-code helps DevOps team to complete development and deployment processes with a reduced timeline from months to days and hours e.g. Low-Code GUI frameworks will help in reducing the time of self-coding and developers can meet the requirements of operations team on time.

Consistency. Low-code platforms can integrate easily with programming tools that automate application deployment. Besides, they can assist in some of the vital processes in any DevOps culture such as version control, build validation, and quality assurance. Having one platform that can provide all this with 'no' or 'low' code can increase the speed of DevOps teams and, at the same time, reduce the overall complexity. Low-code platforms easily get integrated with the organization's workflow. There is not any risk of disruption involved which makes the teams confident.

Threats to Validity. In this study, a small number of participants (n=12) threaten external validity. We will recruit more participants as our study progresses to improve the quality of the findings and aim for theoretical saturation as performed in [28].

A GT approach does not claim generalization, but rather produces a mid-ranged theory applicable to the contexts studied [11]. Data collected during this study does not represent the whole DevOps community and is limited to practitioners who accepted to participate, i.e., convenience sample. We kept the detail of these participants confidential as per human ethics guidelines governing this study.

Other threat is that there were two participants working as project managers who do not use low-code tools very often, so their perspective is different from those who deal with the low-code platforms in a daily basis. To mitigate this threat, we will recruit more participants. Since all the emerged codes, concepts, and categories (including their related properties and subcategories) came from the real data which were collected directly from the context and finalized by all authors to overcome any bias. The variations and classifications proposed by this study can be extended to apply more widely to other aspects of software process improvement in DevOps. We hope future studies can use, validate, and extend our classification in other aspects of process improvement.

Some participants are self-reporting their experience but other participants deal with the adoption of low-code from different roles. We mitigated this threat by an iterative discussion on a related topic to interrogate it properly [32]. We also shared with them, topics to be discussed before the interview session to be sure about the participant, their organization, and their experience in the use of DevOps and low-code approaches.

6 Conclusions and Future Work

DevOps is increasingly becoming the de-facto paradigm approach however, still we rely on the generation of code that can meet the requirements of continuous integration, continuous deployment, and continuous testing to bring overall agility to software process development. To overcome this challenge, our identified emergent categories describe the concept of merging DevOps with low-code approaches.

This study presents a set of emerging results and a vision that could allow an organization to deploy and manage software products at the same fast rate as development. With low-code, a DevOps team can incorporate developers with almost all competence level because low-code provides an opportunity to tackle programmers' shortages. The rising demands of software projects are something to be worried about but, according to practitioners, the emergence of low-code in DevOps is the right channel that can help teams efficiently meet the market demands.

We believe that our findings and relationships with respect to the SPI manifesto lay the foundation for future studies. We will expand our study to examine various low-code platforms to assess the best approaches in the context of DevOps. We also plan to investigate the productivity and quality of a software project that adopts DevOps combined with low-code.

REFERENCES

[1] J. Díaz, D. López-Fernández, J. Pérez, and Á. González-Prieto, 'Why are many businesses instilling a DevOps culture into their organization?', *Empir Software Eng*, vol. 26, no. 2, p. 25, Mar. 2021, doi: 10.1007/s10664-020-09919-3.

[2] S. Rafi, W. Yu, M. A. Akbar, S. Mahmood, A. Alsanad, and A. Gumaei, 'Readiness model for DevOps implementation in software organizations', *Journal of Software: Evolution and Process*, vol. 33, no. 4, p. e2323, 2021, doi: 10.1002/smr.2323.

[3] M. A. Al Alamin, S. Malakar, G. Uddin, S. Afroz, T. B. Haider, and A. Iqbal, 'An Empirical Study of Developer Discussions on Low-Code Software Development Challenges', in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, May 2021, pp. 46–57. doi: 10.1109/MSR52588.2021.00018.

[4] A. Bucaioni, A. Cicchetti, and F. Ciccozzi, 'Modelling in low-code development: a multi-vocal systematic review', *Softw Syst Model*, Jan. 2022, doi: 10.1007/s10270-021-00964-0.

[5] H. Henriques, H. Lourenço, V. Amaral, and M. Goulão, 'Improving the Developer Experience with a Low-Code Process Modelling Language', in *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, New York, NY, USA, Oct. 2018, pp. 200–210. doi: 10.1145/3239372.3239387.

[6] A. C. Bock and U. Frank, 'Low-Code Platform', *Bus Inf Syst Eng*, vol. 63, no. 6, pp. 733–740, Dec. 2021, doi: 10.1007/s12599-021-00726-8.

[7] R. Koplowitz and J. Rymer, 'The Forrester Wave^{TM}: Low-Code Development Platforms For...' Accessed: Apr. 28, 2022. [Online]. Available: https://www.forrester.com/report/The-Forrester-Wave-LowCode-Development-Platforms-For-ADD-Professionals-Q1-2019/RES144387

[8] P. Vincent, M. Driver, and J. Wong, 'Low-Code Development Technologies Evaluation Guide', 2019. Accessed: Apr. 28, 2022. [Online]. Available: https://www.gartner.com/en/documents/3902331

[9] FACT.MR, 'Low Code Development Industry is Projected to Achieve a Global Market Size of US$ 187 Bn by 2032, Currently US Accounts For the Largest Market Share in the World', *GlobeNewswire News Room*, Mar. 09, 2022. https://www.globenewswire.com/news-release/2022/03/09/2400432/0/en/Low-Code-Development-Industry-is-Projected-to-Achieve-a-Global-Market-Size-of-US-187-Bn-by-2032-Currently-US-Accounts-For-the-Largest-Market-Share-in-the-World.html.

[10] C. Jawale, 'How low-code can fit into the DevOps culture'. https://www.opcito.com/blogs/how-low-code-can-fit-into-the-devops-culture (accessed Apr. 28, 2022).

[11] B. G. Glaser, A. L. Strauss, and E. Strutzel, 'The discovery of grounded theory; strategies for qualitative research.', *Nursing research*, vol. 17, no. 4, p. 364, 1968.

[12] J. Smeds, K. Nybom, and I. Porres, 'DevOps: A Definition and Perceived Adoption Impediments', in *Agile Processes in Software Engineering and Extreme Programming*, Cham, 2015, pp. 166–177. doi: 10.1007/978-3-319-18612-2_14.

[13] C. Young and H. Terashima, 'How Did We Adapt Agile Processes to Our Distributed Development?', in *Agile 2008 Conference*, Aug. 2008, pp. 304–309. doi: 10.1109/Agile.2008.7.

[14] F. M. A. Erich, C. Amrit, and M. Daneva, 'A qualitative study of DevOps usage in practice', *Journal of Software: Evolution and Process*, vol. 29, no. 6, p. e1885, 2017, doi: 10.1002/smr.1885.

[15] A. Dyck, R. Penners, and H. Lichter, 'Towards Definitions for Release Engineering and DevOps', in *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, May 2015, pp. 3–3. doi: 10.1109/RELENG.2015.10.

[16] F. Erich, C. Amrit, and M. Daneva, 'Cooperation between information system development and operations: a literature review', in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, New York, NY, USA, Sep. 2014, p. 1. doi: 10.1145/2652524.2652598.

[17] M. Valdes Faura, 'Low-Code and DevOps: Friends or Foes?', *DevOps.com*, Apr. 08, 2021. https://devops.com/low-code-and-devops-friends-or-foes/ (accessed Apr. 28, 2022).

[18] D. A. van der Burgh, ‘A Readiness self-assessment model for Low-code development enabled devops’, Eindhoven University of Technology, Eindhoven, 2019.

[19] J. Philippe, H. Coullon, M. Tisi, and G. Sunyé, ‘Towards transparent combination of model management execution strategies for low-code development platforms’, in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, New York, NY, USA, 2020, pp. 1–10. https://doi.org/10.1145/3417990.3420206

[20] M. Tisi *et al.*, ‘Lowcomote: Training the next generation of experts in scalable low-code engineering platforms’, 2019.

[21] J. Pries-Heje and J. Johansen, ‘SPI Manifesto’. 2010. [Online]. Available: http://www.madebydelta.com/imported/images/DELTA_Web/documents/Ax/SPI_Manifesto_A.1.2.2010.pdf

[22] A. A. Khan and M. Shameem, ‘Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process’, *Journal of Software: Evolution and Process*, vol. 32, no. 10, p. e2263, 2020, doi: 10.1002/smr.2263.

[23] A. Strauss and J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory.* London: SAGE, 1998.

[24] R. Hoda, ‘Socio-Technical Grounded Theory for Software Engineering’, *IEEE Transactions on Software Engineering*, pp. 1–1, 2021, doi: 10.1109/TSE.2021.3106280.

[25] S. Rafi, W. Yu, and M. A. Akbar, ‘Towards a Hypothetical Framework to Secure DevOps Adoption: Grounded Theory Approach’, in *Proceedings of the Evaluation and Assessment in Software Engineering*, New York, NY, USA, Abril 2020, pp. 457–462. doi: 10.1145/3383219.3383285.

[26] B. G. Glaser, *Theoretical sensitivity: advances in the methodology of grounded theory*. Sociology Press, 1978.

[27] K. Charmaz, *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*, 1st edition. London ; Thousand Oaks, Calif: SAGE Publications Ltd, 2006.

[28] K.J.,Stol, P. Ralph and B. Fitzgerald. Grounded theory in software engineering research: a critical review and guidelines. In *Proceedings of the 38th International conference on software engineering* (pp. 120-131) May, 2016.

[29] J. W. Creswell, *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*, 4th edition. Thousand Oaks: SAGE Publications, Inc, 2014.

[30] R. K. Yin, ‘Validity and generalization in future case study evaluations’, *Evaluation*, vol. 19, no. 3, pp. 321–332, Jul. 2013, doi: 10.1177/1356389013497081.

[31] J. Metrôlho, F. Ribeiro, and R. Araújo, ‘A strategy for facing new employability trends using a low-code development platform’, presented at the 14th International Technology, Education and Development Conference, 2020.

[32] N. James and H. Busher, ‘Credibility, authenticity and voice: dilemmas in online interviewing’, *Qualitative Research*, vol. 6, no. 3, pp. 403–420, Aug. 2006, doi: 10.1177/1468794106065010.