Factors that affect the utilization of low-code development platforms: survey study

Hana A. ALSAADI, Dhefaf T. RADAIN, Maysoon M. ALZAHRANI, Wahj F. ALSHAMMARI, Dimah ALAHMADI, Bahjat FAKIEH*

Information Systems Dept., King Abdulaziz University, Jeddah, Saudi Arabia

hralsaadi@stu.kau.edu.sa, dradain@kau.edu.sa, mahmedalzahrani0006@stu.kau.edu.sa, wsalehalshamary@stu.kau.edu.sa, dalahmadi@kau.edu.sa, BFakieh@kau.edu.sa

***Corresponding author:**

Bahjat FAKIEH
BFakieh@kau.edu.sa

Abstract: Low-code development platforms were introduced as a solution to the dilemma of the gap between the high demand for applications and the low number of developers available to meet this demand. The low-code development platforms help the developers to build fast, efficient, and scalable applications with a minimal need for coding, which introduced the concept of citizen developers in the field of application development.

This study explores the factors that attract the developers and programmers to utilize LCDP and discusses some of the problems and challenges that prevent other programmers and developers from using it. An online survey was conducted in Saudi Arabia among 49 respondents of professional developers from different departments of Information Technology in several kinds of businesses as well as students from the Computing and Information Technology faculties to understand the developers' motivations to adopt low-code development platforms. A total of 19 respondents were using LCDP, while the other 30 were not. The paper also highlights the reasons why some developers are not interested in moving toward low-code and commit to the traditional programming approach.

The results of this study explain the factors and advantages that prompt developers to use LCDP and identifies the concerns that prevent them from using it. Reducing development time is considered as an advantage by the majority of the sample that used LCDP. Additionally, the Minimum coding needed reduces the development time and make the application development much easier. On the other hand, some of the reasons for not using LCDP by the developers were the low level of scalability in these platforms, and a lack of knowledge about these platforms and how to deal with them.

Keywords: factor, low-code, platforms, traditional programming, utilization.

1. Introduction

The need for mobile and web applications is increasing because humans are very dependent on them in their daily lives (OutSystems, 2019). This increasing demand boosted the sales of the smartphones which as a result caused a continuous need for a high number of efficient mobile applications (Chang & Ko, 2017). Also, the digital transformation that happened in the business environment required automation in several aspects of the business. Digital transformation starts by turning documents from paper-based into digital forms to convert the business manual processes into digital processes. As a result, a noticeable reduction in the need for a human workforce appeared because automation uses reliable software with fewer errors and less operational cost in the long term (Metrôlho, Ribeiro & Araujo, 2020). This transformation helped businesses to gain several benefits such as improved agility/accelerated innovation, reduced costs/improved efficiency, and growth in new markets (OutSystems, 2019). Three different development approaches assisted the acceleration of this digital transformation. Those approaches are the traditional, low-code and no-code development approaches. It is essential to realize the differences among them since they vary from each other in several features. Some of those properties are the size of the implementation team, the required capabilities, the prototype and the product development time, cost, modification, and investment risk (Moskal, 2021). Traditional development is the technology of coding which is completely adjusted to the task structure and mechanisms. It aims to implement the desired function and shows excellent efficiency in the software (Chang & Ko, 2017) by a team that have certain knowledge in programming (Sahinaslan,

https://doi.org/10.33436/v31i3y202110

Sahinaslan & Sabancıoglu, 2021). Low-code is considered the fourth generation of programming languages (Moskal, 2021). It is a visual integrated development environment that is based on automatic code generation and model-driven design (Sahinaslan et al., 2021) and defined as "a software that provides a development environ-ment used to create application software through graphical user interfaces and configuration instead of traditional hand-coded computer programming" (Metrôlho, Araújo, Ribeiro, & Castela, 2019; Shaikh). Finally, no-code development which is considered as a part of the LCDP market; but it focuses on building applications without writing any line of code (Bloomberg, 2017; Vincent, et al., 2019), where only some text entry is needed for expressions, formulas and the other requirements for the application development that are held throughout the visual modelling and configuration (Vincent et al., 2019). The functionalities of the developed application using the no-code approach is fairly limited comparing to the other approaches (Bloomberg, 2017). It is also considered by Gartner as a part of the low-code approach (Vincent et al., 2019) as both were developed to overcome the limitation of IT staff and the high demand on IT departments (Metrôlho et al., 2020).

The main focus of this study is the low-code development platforms (LCDP) where businesses shifted to it as a solution to the shortage of technical software developers which is becoming very powerful (Silva, et al., 2020; Woo, 2020). By the year 2024, 65% of the applications are expected to be developed by LCDPs as they support the development of different use cases such as reporting, analysis event processing, the user interfaces, data services, and business logic (Rymer et al., 2019; Vincent et al., 2019). Because LCDP uses the concept of drag-and-drop, it provides the option for non-technical developers, who are called citizen developers, to quickly generate and deliver the required applications with minimal effort. That provides the opportunity for professional developers to focus on the critical application development operations such as installation, configuration, and training of the system (Waszkowski, 2019). Some of the leaders in LCDP are OutSystems, Salesforce, Kony, Microsoft, Appian, and Mendix (Metrôlho et al., 2020).

The debate over applications development platforms is rising. Some people are saying that the business should follow the traditional software development life cycle. On the other hand, low-code development pursuers are providing their argument about why low-code is a better approach in developing applications. The most important feature that comes with adopting low-code is the ability to develop applications by novice and non-developers (Chang & Ko, 2017). However, this makes it difficult to update the application to satisfy the user's needs (Woo, 2020). Meanwhile, with traditional software development, though the application should be developed by a professional developer, updating and maintaining the application is not a difficult job to do (Majanoja, Avikainen & Leppänen, 2017).

According to Gartner, over 65% of application development by the year 2024 will be done using LCDP (Margaria & Steffen, 2020). The COVID-19 Engagement Portal is an example of the applications that were developed with LCDP. This is an application made for New York City officials who wanted to track how the Coronavirus (COVID-19) had been raging through the city, so that they could efficiently provide services to those in need. To accomplish this, they built this online portal to collect information about individuals with COVID-19 and others in contact with them. Speed was paramount, and the application was up and running in three days without any code written (Woo, 2020).

This paper highlights the concept of LCDP in detail. It aims to explore the factors that make programmers and developers use LCDP. Moreover, it seeks to figure out the reasons that make them prefer to use the traditional programming approach over LCDP. These factors are defined through a survey study. The sample size of the conducted study consists of 49 programmers/ developers. The highest advantage of LCDP, as mentioned in the results of this study, was reducing the time required to develop an application. One of the major factors that led developers and programmers to prefer using traditional programming approaches over LCDP was the limited level of scalability in LCDP.

The structure of this paper is as follows. Section 2 discusses a literature review on some research about LCDP. Section 3 presents the research methodology. Section 4 discusses the data analysis and results. Section 5 discusses the results. Section 6 contains the conclusion and future work.

2. Literature review

The traditional programming approach and low-code development platform approach will be discussed from different perspectives. These perspectives include time, cost, effort, security, and the level of needed experience.

In the traditional programming approach, the programmers must write a huge amount of code to develop an application. In some situations, the programmer will be required to write the code from scratch (Sanchis, et. al., 2020). This causes a lot of costs in terms of time, money, and effort (Sanchis et al., 2020; Sattar, 2018). Some researchers indicate that the time spent by programmers to write the code is the most complex part of the development process. That is why this approach is considered to be time-consuming (Strømsted, Marquard & Heuck, 2018; Vikebø & Sydvold, 2019). Moreover, the developer needs to have good experience and knowledge in the development activities and programming languages to be able to develop an application (Metrôlho et al., 2019; Sanchis et al., 2020; Strømsted et al., 2018).

On the other hand, Low-Code Development Platforms (LCDP) enable fast development and delivery of applications (Sanchis et al., 2020; Strømsted et al., 2018; Vikebø & Sydvold, 2019). Not only professional developers would take the advantage of developing applications, beginners became able to develop an application without needing experience or knowledge about any programming languages or complex engineering activities (Metrôlho et al., 2019; Strømsted et al., 2018). The application will be developed by dragging and dropping the components and the code will be generated in the background (Adrian, Hinrichsen & Nikolenko, 2020). The literature illustrates that the code generated by LCDP will be difficult to understand because it is written without comments, the used variables may have vague names and some security issues (Woo, 2020). However, other literature has a different viewpoint. It states that using LCDP protects privacy because the end-user or organization's employees will develop the application instead of outsourcing that work (Sanchis et al., 2020).

This section discusses the following. Section 2.1 will be about the significance of low-code development. Section 2.2 is about low-code development platforms including examples and details about them. Section 2.3 will discuss the factors that affect the adoption of LCDP.

2.1. The significance of low-code development

The rapid changes in the market requirements necessitate rapid and flexible responses from companies and organizations (Sanchis et al., 2020; Woo, 2020). The researchers illustrated that software solution development helps companies and organizations to meet the changes in the market in an effective way and enhance their digital transformation (Sanchis et al., 2020). This growing demand for applications and software has increased the need for programmers, developers, and IT specialists (Metrôlho et al., 2019). As analyzed and expected by Gartner, this high demand in enterprises will be growing around five times faster than the ability of IT professionals to deliver applications by 2021 (Chang & Ko, 2017; Hyun, 2019). Many industry analysts, researchers, and corporate executives confirm that it is becoming difficult for IT departments to fulfill the businesses' needs for the fast delivery of efficient applications, as Young-Hyun Chang and Chang-Bae Ko stated (Chang & Ko, 2017). Moreover, higher education institutions have become unable to cover the shortage of Information and Communication Technology (ICT) professionals required by businesses to meet their high demands for applications (Metrôlho et al., 2020). To bridge the gap between the high demand, the lack of ICT professionals, and the intensive workload in IT departments, the usage of Low-Code Development Platforms (LCDP) has become more important (Adrian et al., 2020).

The main goal of the tools and features of LCDP, as illustrated by the researchers, is to give companies and organizations the ability to develop software and applications without complex coding experience (Sanchis et al., 2020). LCDP delivers user experience through mobile and web applications, while including complex forms, single-page apps, and page navigation. In addition, reporting, data management, collaboration, and workflow automation tools and features are also provided by LCDP (Rymer et al., 2019). The growing use of LCDP decreases the workload on IT department professionals and allows them to focus on complex programming tasks (Adrian et al., 2020). This view would allow non-professional developers to develop and deploy less complex applications quickly and with less effort by harnessing LCDP (Adrian et al., 2020).

2.2. Low-Code Development Platforms

As the literature stated, Forrester Research coined the term "low-code" in 2014 (Sanchis et al., 2020), LCDP is a visual integrated development environment (IDE) that enables non-professional users to create, develop, and deploy applications without the need for experience in programming. Instead, these platforms use declarative, high-level programming abstractions (Hyun, 2019; Margaria & Steffen, 2020; Metrôlho et al., 2019). LCDP is also known as end-user software engineering, End-User Programming (EUP), and meta-design (Silva et al., 2020).

The market of LCDP offers tens of platforms, each of which targets different domains, differentiate, gain more competitive advantages, and attract more customers. Different vendors such as Oracle, Microsoft, Alibaba, and Salesforce have adapted a specific type for their low-code development platforms (Woo, 2020).

Adapting artificial intelligence (AI) is one of the competitive advantages between those vendors. The main goal of this is to improve the users' experience (Appian, Announcing the Latest Version of the Appian Low-code Automation Platform). Moreover, one of those vendors, OutSystems, took care of some of the major concerns that limit the use of low-code development platforms to gain more customers. Two of these concerns are scalability and security (Oltrogge et al., 2018; Warren, 2018).

OutSystems is the market founder of LCDP (OutSystems, 2019) More than 1200 companies in 52 countries use it. It has a network of more than 250 partners (Metrôlho et al., 2019). The focus of OutSystems is dedicated to develop enterprise applications that are used for automating the core business processes for agile and continuous delivery (OutSystems, 2019) such as billing systems, ERPs, CRMs, additions for current ERP solutions, business intelligence and dashboards (Sahay, et al., 2020). The OutSystems platform is designed to help developers deliver the applications efficiently and quickly, ensuring that they are enterprise-grade. AI and ML are used for providing recommendations, automation, and validation of the developed application. This enhances the delivery of the application with high quality and at up to 100 times faster. The developed applications are resilient, secure, and built to scale (OutSystems). It expands its features to new fields of business applications. This includes real-time data and artificial intelligence (OutSystems, 2019). Powerful automation, visual development, and AI assistance are some of the tools the developers can use to build and deploy applications which range from regular consumer applications to critical systems applications.

Salesforce offers another well-known cloud based LCDP. It is considered among the largest low-code platform (OutSystems, 2019) that focuses on the customer related applications (Vincent et al., 2019). It uses the Lightning Framework to create apps faster at a lower cost. The customization of themes, colors, and everything related to branding is done with simple clicks. Moreover, the components made by Salesforce are easy to reuse without compromising the customization (Salesforce).

A very well-known low-code platform is Mendix (OutSystems, 2019). Reports by analysts like Gartner, IBM, and SAP recognized Mendix as a leading low-code development platform. The platform is designed to achieve rapid development of applications (Mew & Field, 2018) while focusing on the collaboration between business and IT to improve the business logic (Vincent et al., 2019). It has a flexible, open architecture that combines collaboration, speed, and control

(mendix). It separates the design environment between citizen developers and professional developers to optimize collaboration across the business departments (Rymer et al., 2019). For the citizen developers, it is called Mendix Studio, while Mendix Studio Pro (which is a more robust environment) is for professional developers (Vincent et al., 2020). In each application, there is a dedicated space for the project where the collaboration between the developers and the stakeholders takes place. This feature enables the stakeholders to participate and innovate faster, which makes the application more successful (mendix; Mew & Field, 2018). Mendix has leading features in AI and machine-learning features for development (OutSystems, 2019). The use of AI and ML for real-time recommendations and error checking (Haan, 2018) makes the platform suitable for complex applications (Vincent et al., 2019). AI assistant in Mendix acts as a world-class for developers. It guides them to the next step in the development process and checks for errors, inconsistencies, quality, maintainability, and scalability in the developed application. When there is an error, the developer will receive feedback about it and where is it located (Haan, 2018).

One of the experts and most recognized leaders in low-code platforms and automation is Appian (Appian) (Sahay et al., 2020). Although it is smaller than many of its competitors, Appian has several government agencies and business customers that run its platform. Appian's LCDP focuses on the complex processes of business and applications that need high level automation and analysis (Vincent et al., 2019). It specializes in creating mobile and web applications through personalization via intranet using a decision engine that helps in designing complex applications (Sahay et al., 2020), and as a result it is more suitable for professional developers. It focuses on the complex processes that require sophisticated automation and end-to-end case management by offering a stack of low-code tools for automation that handle the complexity of the workflow. Visual tools are available to build applications quickly (Vincent et al., 2020). It is secure, reliable, and scalable to support the complexity of the application being developed (Appian). The Appian platform speeds up the application development process and improves the productivity of ICT professionals and citizen developers by using AI-Guide. It uses the machine learning concept to give the user some recommendations about the next step to take in the development process (Appian, Announcing the Latest Version of the Appian Low-code Automation Platform).

Other vendors of LCDP are Microsoft PowerApps (Sahay et al., 2020) that could be used in several use cases, but considered much more stronger than Microsoft Office 365 (Vincent et al., 2019). It is integrated with many other services of Microsoft such as Microsoft Azure, MS. Teams, MS. Excel, and Kissflow; which generates and adjusts automated business applications. It focuses on small applications that target human-centered workflow such as buying requests, sales reviews, purchase collections, sales channel, and software directory (Sahay et al., 2020)

The vendors that adopted AI in their platforms gained a competitive advantage over others. The platforms work by understanding the user's behavior and start giving suggestions and recommendations regarding application development. Some of these providers integrate AI assistants in the platform; his is done by using ML algorithms and a huge set of historical data to provide better advice and suggestions to the end-user (Woo, 2020).

2.3. Factors affecting the adoption of LCDP

LCDP provides useful solutions in the process of automating and speeding up the delivery of applications. This enables customers to have a high rate of growth. However, LCDP also introduces the risk of small developers selling their applications with no technical management to customers who do not know how these applications were developed and if these apps meet their demands (Sanchis et al., 2020; Woo, 2020).

According to Forster study (Rymer & Appian, 2017) that included 41 participations, the challenges that are faced by the organization while using the traditional programming approach for building custom applications are as the following:

- Meeting business requirements on time.
- Limited flexibility.

- Need long time to update the apps.
- Limited qualified developers.
- Unsatisfied customers.
- Costly.
- Low quality.

Based on the respondents, the adoption of low-code significantly improved these issues while most of the developed apps were enterprise wide apps (Rymer & Appian, 2017).

In 2019, OutSystems (OutSystems, 2019) conducted a survey on a collection of more than 3300 IT professionals from six countries (OutSystems, 2019). The factors that affect the organizations' decisions to utilize LCDP are defined because of this study. Some of these reasons are stated by the OutSystems report, "The State of Application Development" (OutSystems, 2019), as follows:

- Increasing digital innovation and transformation.
- Protecting the business from the change that occurs over time as existing customers are lost and new customers are added. This is called Technology Churn.
- Increasing responsiveness to the business.
- Giving citizen developers the ability to improve internal processes.
- Reducing the dependency on employees with technical skills and IT professionals.

On the other hand, some organizations prefer to use traditional development platforms over LCDP. This may be due to some of the reasons listed below, according to the OutSystems report (OutSystems, 2019):

- The shortage in knowledge about low-code platforms.
- Concerns about "lock-in" with a vendor of low-code.
- Lack of belief that LCDP provides the ability to develop the needed types of applications.
- Concerns about the security of the developed applications.
- Concerns about the scalability of the developed applications.

According to Forrester report, the applications of the business processes are ranked at the highest priority for most of the low-code vendors. Very few vendors invest in security certificates such as HIPPA, FedRAMP and SOC 2. The others are relying on their partners infrastructure's certificates (Rymer & Appian, 2017).

Other factors that affect the adoption of LCDP are discussed in the literature of "Low-Code as Enabler of Digital Transformation" (Sanchis et al., 2020; Woo, 2020):

Privacy: The application development tasks are performed internally by technical or non-technical staff from the organization and not by third parties, which increases confidentiality. (OutSystems, 2019).

Rapidity: The users will only have to configure the application visually and make the necessary adjustments in the development process instead of hand-coding them. This is because the development of the main part of the code is done (OutSystems, 2019). Because the development time is reduced, the delivery time of the apps is reduced as well (Sanchis et al., 2020; Woo, 2020).

Cost reduction: The cost of the application development is reduced because the time of the development has been decreased greatly. This is true whether the app is being developed by the company or by a third party (Sanchis et al., 2020; Woo, 2020).

Simplicity: The app development process is simplified because the apps are not built from scratch. This enables the developers to focus on customizing the application to meet the user's needs (Sanchis et al., 2020; Woo, 2020).

Maintainability: To guarantee that the offered services meets the business requirements, the maintenance of the application is vital. There is a need to make very quick changes in what has been developed already. In LCDP, there is not much to change because the essential goal of the LCDP is to offer minimum coding (OutSystems, 2019).

Involvement of business profile: Users of any application can become developers as they know the business needs at the best level. Moreover, the development environment in these platforms provides a simple and intuitive GUI to help these end-users in the application development (Sanchis et al., 2020; Woo, 2020).

Minimization of inconsistent or unstable requirements: Some requirement conflicts might appear in the application development process. If any changes occurred in the requirements, these would impact the app design. However, in low code, the developers are quickly building minimum workable products to validate the ideas and the requirements from the customer. This helps to prevent wasting resources on functionalities and features that customers may not need (Sanchis et al., 2020; Woo, 2020).

Another view of the literature "Low-Code as Enabler of Digital Transformation" (Sanchis et al., 2020; Woo, 2020) highlighted some main limitations for using low-code development platforms:

Scalability: LCDPs are designed mainly to address small apps. Large-scale apps are covered in some platforms.

Fragmentation: Different low-code development models can be defined depending on the customers and their needed programming model.

Software-only systems: While citizen developers have no experience in programming, they have a lot of experience in other engineering areas. These experiences and knowledge should be used in app development.

The main factors that affect the adoption of LCDP and that are shown in the lecture review are summarized in Table 1.

Table 1. Factors that affect LCDP adoption

Reasons for the adoption of LCDP	Increasing digital transformation
	Rapid development
	Increasing responsiveness to the business
	Information privacy
	Cost reduction
	Simplicity
	Maintainability
	Involvement of business profile
	Minimization of inconsistencies
Factors preventing LCDP adoption	Lack of knowledge about LCDP
	Vendor lock-in
	Distrust of the LCDP abilities
	Scalability concerns
	Security concerns
	Fragmentation
	Software-only systems

3. Research methodology

The goal of this research is to explore the factors that affect the developers' decision to adopt either low-code or traditional programming approach. A significant part of the research is to understand how the developers view each of these approaches and what difficulties and challenges are faced while using LCDP. In this section, the research methodology will be discussed by describing how the data was collected using a survey, in section 3.1. The survey design is discussed in section 2.2. Finally, data analysis and results are discussed in section 2.3.

3.1. Survey design

The survey questions were designed to explore the factors that affect the developers' decision to use low-code or the traditional programming approach in the development of their applications. The benefits and challenges that they face while using LCDP are also discussed. Finally, developers were asked for their opinions about the performance and quality of the developed applications using either approach. The survey was designed with 23 questions. The questions were categorized into 3 types. Ten of them were open-ended questions, 10 were closed-ended questions, and 3 were based on the Likert scale. At the beginning of the survey, the participants were asked if they had ever used low-code development platforms. This was to determine their knowledge of low-code development platforms and ensure the validity of the information that they provided. Moreover, understanding the factors that motivated or prevented the participants from using LCDP is an important part of the survey.

The survey has 3 sections. The first section was designed to collect general information about the participants. Assessment was done for the participants to understand their level of experience in programming generally and in low-code development platforms in specific. The questions were about their level of programming experience, how much knowledge the developer had regarding LCDP, and if the developer had ever used LCDP before.

Based on the answer to the last question in the first section, the participant was directed to one of the next two sections. Section number 2 provided questions to highlight the reasons that prevented the developers from using LCDP. Meanwhile, the questions in section 3 were for the developers who had used LCDP.

Section 2 was designed with five questions. The questions started with the reasons that prevented the developer from using low-code development platforms. After that, a set of questions was provided to understand the developer's opinion about LCDP. The questions were about the performance and quality of the developed application and the speed of the development and delivery process when using LCDP. Finally, a question was provided to figure out the developer's future intention to use low-code development platforms.

Section 3 contained 15 questions. The questions in this section were directed toward the factors that affected the developer's decision to adopt LCDP, as well as the challenges faced and benefits gained from using it. The section had two categories of questions. The first category was to determine the used LCDP in the development process, while the second category was to compare LCDP to the traditional programming approach from the participant's point of the view.

In the first category, the developer was asked about which low-code development platform had been used and the type of application developed. Next, the developer was asked about how they benefitted from LCDP when creating the application and the chances of their reusing or recommending LCDP and why. Finally, an open-ended question was provided so that the developers could indicate the challenges and problems they faced during the development process of the application using LCDP.

The second category of this section aimed to compare LCDP to traditional programming from the developer's point of view. It started with two questions that sought to determine the developer's assessment of the performance and quality of the applications developed using LCDP compared to those developed with traditional programming and why. Next, the questions focused

on the code. The developer was asked about the amount of code that should be written and the process of black box debugging when using LCDP as compared to traditional programming. The next set of questions explored the developer's opinion about how LCDP would work with the complex application and how many problems would be faced in the integration process and why. The flowchart in Figure 1 describes the flow of the survey questions.

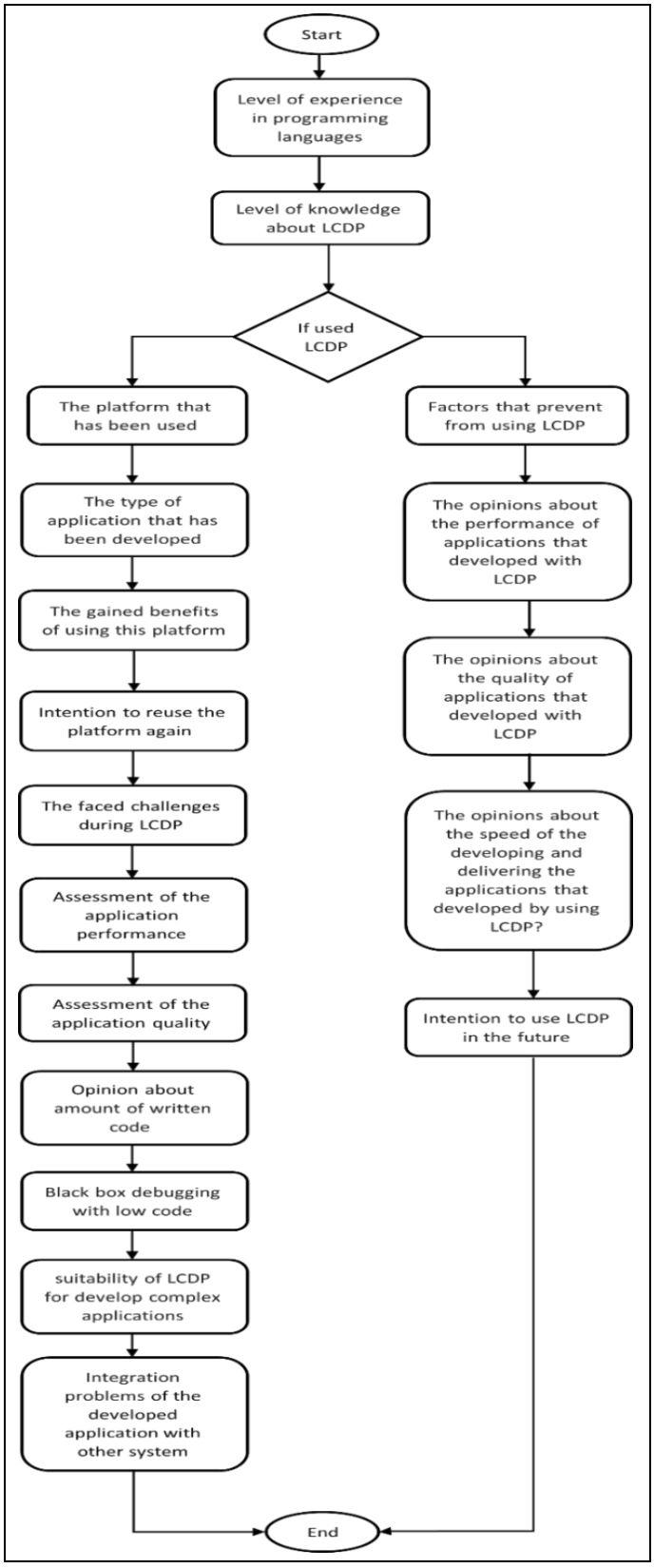

Figure 1. Survey design flow chart diagram

3.2. Data collection

For defining and studying the factors that affect the adoption of low-code development platforms or traditional programming, the researchers conducted a survey. The survey targeted mainly the developers and programmers who had used different low-code development platforms.

The goal of the conducted survey was to reach a sample of at least 20 developers, keeping in mind that the participants would have different levels of knowledge and experience in development with low-code as well as traditional programming languages. The type of collected data was quantitative. To achieve the goal, the conducted survey targeted 2 categories of developers: students and professionals. The data of the first category were collected from the Faculty of Computing and Information Technology students, while the second category of professionals included developers and programmers from different departments of Information Technology in several kinds of businesses. Some of them were from the IT Deanship at King Abdulaziz University and the Deanship of Distance Learning.

4. Data analysis and results

At the end of the conducted survey, the sample reached 49 responses from both categories of students and professional developers. Before processing with data analysis, the collected data were prepared by removing the missing and insufficient entries. After that, the data were organized, and the related data were grouped into categories. There were 3 main categories studied by the researchers. The first one was the added benefits of using LCDP, how the platforms helped in the development process, and the challenges faced while using LCDP. The second one was related to the factors preventing the use of LCDP. The last category was about compression between LCDP and traditional programming approaches. Another analysis for the data was done based on the configured relationships between one dependent variable with other independent variables from the set of collected data. The researchers started to figure out the relationships by using multiple regression tests. The significant value for the multiple regression was 0.05. The first configured relationship was between the use of low-code (the dependent variable) and the knowledge about low-code and the programming experience (independent variables). The second relationship was between the intention to use low-code (dependent variable) and the performance, quality, and speed of the applications developed with LCDP (independent variables). The qualitative data were analyzed using statistical software.

Table 2 shows the distribution of knowledge and experience in LCDP and programming language variables among the sample of the survey. Most of the sample had an intermediate level of knowledge in programming languages (30/49, 61.22%). For the level of knowledge of LCDP, the sample that had a limited level of knowledge was (19/49, 38.78%), and the sample that did not have any knowledge was (11/49, 22.45%). Most of the sample did not use LCDP, (30/49, 61.22%).

Table 2. Level of knowledge and experience in LCDP and programming languages (N=49)

Variable	Frequency & Percentage n (%)
Level of knowledge in programming languages	
Expert	13 (26.53)
Intermediate	30 (61.22)
Beginner	5 (10.20)
No prior experience	1 (2.04)
Level of knowledge about LCDP	
Excellent knowledge	4 (8.16)
Good knowledge	15 (30.61)
Limited knowledge	19 (38.78)
No prior experience	11 (22.45)
Using LCDP	
Yes	19 (38.78)
No	30 (61.22)

4.1. Results related to those who have used LCDP

The results in Figure 2 show that most of the sample, 12 participants representing 63.16% have used LCDP developed E-commerce applications and websites. Another 3 participants (15.79%) used it in creating organization systems. Only 2 participants representing 10.52% used it to develop healthcare applications, while another 2 used it in other different type of applications.

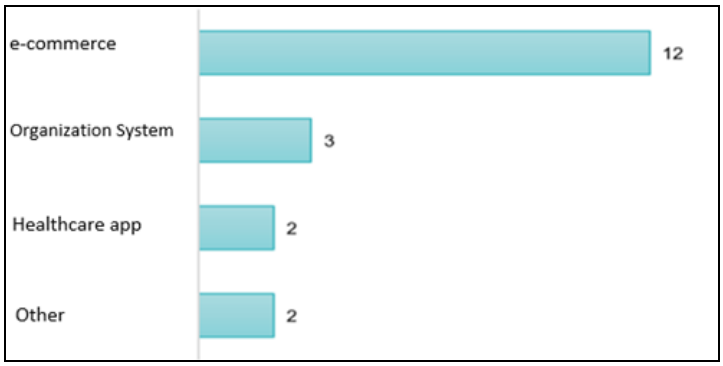

Figure 2. Type of the developed application by LCDP

Most of the sample considered two main features, which are the ease of development and the significant time saving of the development, nominated by 18 participants representing 94.74% for each feature. The least-considered feature with 26.32% was the suggestions by the system to the next step. The considered benefits of using LCDP are shown in Figure 3.

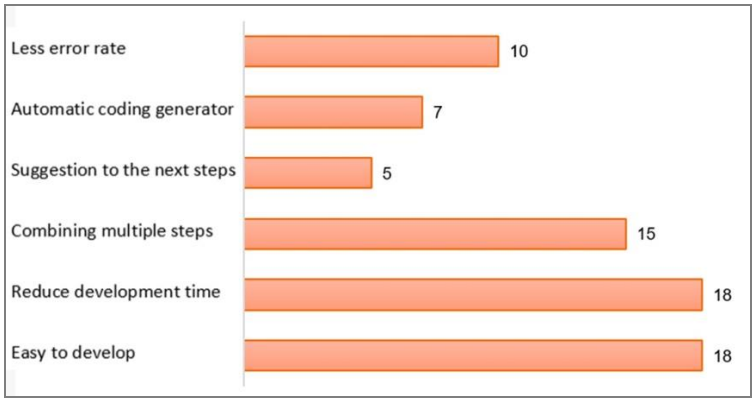

Figure 3. The gained benefits of using LCDP

Most of the sample considered one main challenge, which is the difficulty of customizing some built-in functions in the platforms; this was nominated by 6 participants representing 31.58% of the sample. The least-considered challenge was the dependency on network speed. The considered challenges of using LCDP are shown in Figure 4.

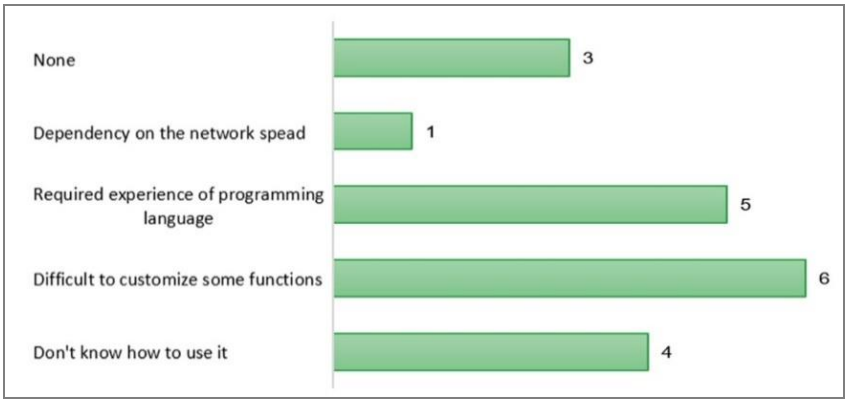

Figure 4. The faced challenges during the use of LCDP

Table 3 illustrates the questions of the sample using LCDP and their response rate for each question. It is noticeable that 52.63% of the sample found that the quality of the applications developed with LCDP is equal to the quality of those developed using traditional programming approaches. Furthermore, 57.89% of the sample believe that it is better to write less code when developing applications with LCDP. On the opposite side, 47.39% find that LCDP is not suitable to develop complex applications, while 26.32% of the sample think that black box debugging with LCDP is an issue for the developer. Meanwhile, 52.36% of the sample did not face any problems in the integration of the applications developed with LCDP to other systems.

Table 3. Response rate of the sample using LCDP (N=19)

Question	**Agree rate (n)**	**Disagree rate (n)**	**Neutral (n)**
LCDP develops apps with better quality than traditional programming.	36.84% (7)	10.53% (2)	52.63% (10)
It is better to write less code when developing apps with LCDP.	57.89% (11)	10.53% (2)	31.58% (6)
LCDP is not suitable to develop complex systems.	42.10% (8)	47.39% (9)	10.53% (2)
Black box debugging in low-code is not an issue for the developer.	36.84% (7)	26.32% (5)	36.84% (7)
It is hard to integrate the developed apps with LCDP to other systems.	10.53% (2)	52.63% (10)	36.84% (7)

Is the decision to use LCDP affected by the developer's programming experience and knowledge about LCDP? To answer this question, multiple regression is used.

- The null hypothesis: There is no relationship between the decision to use LCDP and the programming experience and knowledge about LCDP.
- Alternative hypothesis: The decision about using LCDP is affected by programming experience and knowledge about LCDP.

Table 4 shows the result of multiple regression regarding the relationship between using LCDP and the developer's programming experience and knowledge about LCDP. According to the results, both of the obtained P value for LCDP knowledge and the programming experience are less than the significant value ($\alpha = 0.05$). Thus, the null hypothesis is rejected. There is a relationship between the decision to use LCDP and knowledge about it and the programming experience.

Table 4. Multiple regression of using LCDP

Variable	**F value (df)**	**P value**
Programming experience	7.055 (2,46)	0.0478
Knowledge about LCDP	7.055 (2,46)	0.0006

Table 5 shows the relationship between using LCDP to develop large and complex systems and the performance, quality, amount of code to be written, black box debugging, and integration problems of the apps developed using LCDP. The conducted test shows that there is a relationship between the black box debugging in LCDP and the intention to use LCDP for developing complex systems because the P value of this variable is less than the significant value ($\alpha = 0.05$). On the other hand, there is no relationship between the intention to use LCDP for developing complex systems and other variables shown in table 5. The null hypothesis is rejected.

Table 5. Multiple regression of using LCDP for complex applications

Variable	F value (df)	P value
Performance	3.386 (5,13)	0.186
Quality	3.386 (5,13)	0.448
Amount of code	3.386 (5,13)	0.798
Black box debugging	3.386 (5,13)	0.042
Integration problems	3.386 (5,13)	0.775

The insignificance in the above results could be due to specific low-code platform the developers used during the application development and due to their personal experience with those platforms, as well as the types of developed applications. As mentioned in Figure 2, 12 out of the 19 participants who utilized LCDP, used it to develop e-commerce solutions such as simple websites that do not show clear differences in some of the tested factors such as quality and performance. The poor significance of affecting the amount of code and the integration issues may be related to the level of knowledge and developers' experience, where most of them were not expert developers and have limited experience in expecting the amount of the required code to develop such applications with the traditional programming approach. Black box debugging was the only factor that has a significant a relationship to LCDP development, and this would be related to the limited knowledge of the participated developers in troubleshooting coding errors. Thus, the black box debugging assesses them to refine the code. However, these results may change if the study was conducted over a larger sample.

4.2. Results related to those who did not use LCDP

Most of the samples considered a problem with LCDP that prevented them from using it, which is the limited level of scalability in LCDP, nominated by 19 participants represents 63.33% of the sample. The least considered problem was that LCDP is not available in the workplace. The considered problems and factors that prevent the use of LCDP are shown in Figure 5.

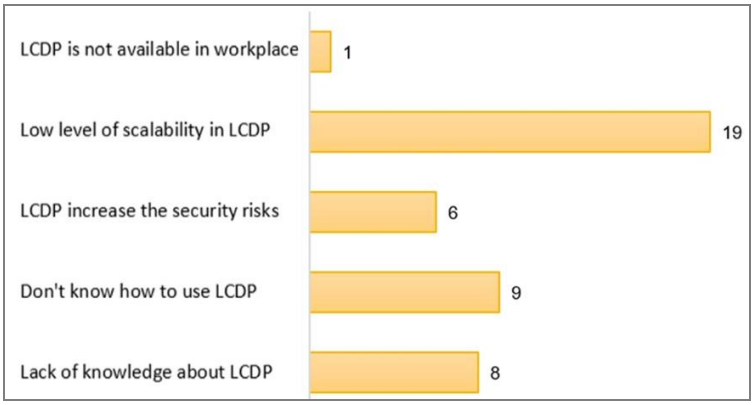

Figure 5. Factors that prevent the use of LCDP

Table 6 displays the questions of the sample that did not use LCDP and their response rate for each. In total, 60% of the sample has an intention to use LCDP in the future. Around 63.33% of the sample know that the use of LCDP will reduce the development time, while about 43.33% of the sample think that the applications developed using LCDP have good quality and performance.

Table 6. Response rate of the sample who does not use LCDP (N=30)

Question	Agree Rate (n)	Disagree Rate (n)
Apps developed by LCDP have good quality.	43.33% (13)	20% (6)
LCDP decreases development time.	63.33% (19)	10% (3)
Intention to use LCDP in the future.	60% (18)	40% (12)

Is the developers' intention to use LCDP in the future affected by their ideas about the quality, performance, and development speed of the LCDP-developed apps? To answer this question, multiple regression is used.

The null-hypothesis: There is no relation between the developer's intention to use LCDP and ideas about the quality, performance, and development speed of the LCDP-developed apps.

Alternative hypothesis: The developer's intention to use LCDP is affected by ideas about the quality, performance, and development speed of the LCDP-developed apps.

Table 7 shows the result of multiple regression regarding the relationship between the intention to use LCDP and the developer's ideas about the quality, performance, and development speed of apps developed with it. According to the results, P values of all factors were higher than the significant value, which is $\alpha = 0.05$. Therefore, the null hypothesis is accepted. There is no relationship between the developers' ideas about the quality, performance, and development speed of apps developed using LCDP and the intention to use it.

Table 7. Multiple regression of intention to use LCDP

Variable	F value (df)	P value
Performance of apps developed by LCDP	0.744 (3,26)	0.427
Quality of apps developed by LCDP	0.744 (3,26)	0.498
Development speed when using LCDP	0.744 (3,26)	0.783

5. Discussion

The results of this study presented that the knowledge level of dealing with low-code platforms and programming experience affects the developer's decision of using LCDP. It also shows the factors that benefit developers and programmers with regard to using LCDP. Some of these benefits were that the application was easy to develop in less time without much coding. Most of the sample who used LCDP had intermediate experience in the programming languages. As a result, around 95% of the sample considered the reduction in the development time as an advantage. This result agrees with the literature that considered the use of deep learning and machine learning in the LCDP as helping to identify the pattern of the user's coding and giving them recommendations which help/aid in accelerating the development time with less errors (Woo, 2020). The participating sample considered the easy development as an advantage as well. The literature supports this point, where more individuals will have the ability to develop programs and give a room for the expert developers to work on the much sophisticated projects. On the other hand, the literature highlighted that the demand for the experts may be reduced, but their experience will still be needed since there will be several projects' requirements that need their experience (Woo, 2020). These results added some extra factors and benefits of using LCDP to those factors defined by the literature (OutSystems, 2019; Sanchis et al., 2020; Woo, 2020).

The major challenge the sample faced was customizing the built-in functions in the low-code platforms in addition to their lack of experience in dealing with programming languages. Also, some of them experienced difficulties while using LCDP because they did not know how to use and deal with these platforms.

For those developers and programmers who did not use LCDP, the major factor that they were concerned about and that prevented them from using these platforms was the limited level of scalability in LCDP. The concerns about scalability exceeded the concerns about security risks. This challenge is also mentioned by some of the literature (OutSystems, 2019; Sanchis et al., 2020; Woo, 2020). The majority of the sample who did not use LCDP had an intention to use these platforms in the future.

6. Conclusion and future work

People are becoming more dependent on technology in all aspects of their lives. The need for mobile applications and other systems is increasing day-by-day. This huge demand is causing a lack of ICT professionals. LCDP provide the opportunity for non-expert individuals, who are called citizen developers, to develop applications without the need for programming experience. Using LCDP helps to develop applications in less time, more easily, and with less coding. A significant part of the development is done by GUI using drag-and-drop components.

This study investigated the factors that lead to utilize LCDP and the challenges of utilizing it. Also, it figured out the factors that prevent some developers or affect their decision about using LCDP. The research targeted professional developers from IT departments in different businesses and students of computing-related departments in Saudi universities. Many programmers and developers preferred LCDP over the traditional programming approach. Some of the factors that attracted them to LCDP were a reduction in app development time, the ease of use, the automatic code generation, the providing of suggestions for the next step, and lower error rates. Yet, developers and programmers mentioned some problems that were faced during the use of LCDP. The highest-rate problem was difficulties in customizing some functions that were built into the platforms in advance.

As with any technology, although LCDP provides a set of benefits for programmers and developers, they face some issues and challenges while using it. They are major concerns that affect the developer’s decisions toward adopting LCDP for application development. In some cases, developers avoid using LCDP and prefer traditional programming. Limited scalability, security risks, and problems integrating the apps developed by LCDP with other systems are some of these challenges.

In addition, more studies and assessments must be done by researchers to find out and suggest appropriate solutions to the issues and challenges of LCDP. Addressing these issues may help to improve the developers' and programmers' experiences with LCDP. Also, this could attract those who do not use LCDP to start using it. The most significant challenges that should be considered are the limited level of scalability for the developed apps and the security issues generated while using LCDP.

Furthermore, the LCDP providers are encouraged to take steps toward improving the developer’s experience while using their platforms. Developers should have more flexibility for customizing the built-in functions in these platforms. Moreover, increasing the level of security of the apps developed with LCDP must be taken into consideration by the providers. Also, the automatically generated code should be more readable, written with evident comments, and use meaningful variables names. This will make it easy to maintain and change the code.

In the future, the survey should be conducted on a larger collection of programmers and developers to determine more factors, benefits, and challenges regarding LCDP.

REFERENCES

1. Adrian, B., Hinrichsen, S. & Nikolenko, A. (2020). *App Development via Low-Code Programming as Part of Modern Industrial Engineering Education.* Paper presented at the International Conference on Applied Human Factors and Ergonomics.
2. Appian. *Appian is a Leader in Gartner Low-Code Magic Quadrant Leader - Again*.
3. Appian. *Announcing the Latest Version of the Appian Low-code Automation Platform*. Retrieved from https://www.appian.com/news/news-item/announcing-the-latest-version-of-the-appian-low-code-automation-platform/
4. Bloomberg, J. (2017). *The Low-Code/No-Code Movement: More Disruptive Than You Realize*. Forbes.
5. Chang, Y.-H. & Ko, C.-B. (2017). *A Study on the Design of Low-Code and No Code Platform for Mobile Application Development*. International journal of advanced smart convergence, 6(4), 50-55.
6. Haan, J. D. (2018). *Introducing AI-Assisted Development to Elevate Low-Code Platforms to the Next Level*. Retrieved from https://www.mendix.com/blog/introducing-ai-assisted-development-to-elevate-low-code-platforms-to-the-next-level/
7. Hyun, C. Y. (2019). *Design and Implementation of a Low-Code/No-Code System*. International journal of advanced smart convergence, *8*(4), 188-193.
8. Majanoja, A.-M., Avikainen, P. & Leppänen, V. (2017). *The impact of agile software development approach on software developers' responsibilities.* Paper presented at the World Conference on Information Systems and Technologies.
9. Margaria, T. & Steffen, B. (2020). *eXtreme Model-Driven Development (XMDD) Technologies as a Hands-On Approach to Software Development Without Coding*. Encyclopedia of Education and Information Technologies, 732-750.
10. Mendix. *Don't write code. Write the future*.
11. Metrôlho, J., Araújo, R., Ribeiro, F. & Castela, N. (2019). *An approach using a low-code platform for retraining professionals to ICT.* Paper presented at the 11th International Conference on Education and New Learning Technologies.
12. Metrôlho, J., Ribeiro, F. R. & Araujo, R. (2020). *A strategy for facing new employability trends using a low-code development platform.* Paper presented at the 14th International Technology, Education and Development Conference.
13. Mew, L. & Field, D. (2018). *A Case Study on Using the Mendix Low Code Platform to support a Project Management Course.* Paper presented at the Proceedings of the EDSIG Conference ISSN.
14. Moskal, M. (2021). *No-Code Application Development on the Example of Logotec App Studio Platform*. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, 11(1), 54-57.
15. Oltrogge, M., Derr, E., Stransky, C., Acar, Y., Fahl, S., Rossow, C., Pellegrino, G., Bugiel, S. & Backes, M. (2018). *The rise of the citizen developer: Assessing the security impact of online app generators.* In 2018 IEEE Symposium on Security and Privacy (SP), 634-647. DOI: 10.1109/SP.2018.00005.
16. OutSystems. OutSystems. *AI Forging the Future of App Dev With AIFusion™*. Retrieved from https://www.outsystems.com/ai/.
17. OutSystems. (2019). *The State of Application Development. Is IT Ready for Disruption? OutSystems*. Retrieved from Boston, MA, USA:
18. Rymer, J. R., & Appian, K. (2017). *The Forrester Wave™: Low-Code Development Platforms For AD&D Pros, Q4 2017*. Cambridge, MA: Forrester Research.
19. Rymer, J. R., Koplowitz, R., Mines, C., Sjoblom, S. & Turley, C. (2019). *The Forrester Wave™: Low-Code Development Platforms For AD&D Professionals, Q1 2019.* Retrieved

from: <https://www.forrester.com/report/The-Forrester-Wave-LowCode-Development-Platforms-For-ADD-Professionals-Q1-2019/RES144387>

20. Sahay, A., Indamutsa, A., Di Ruscio, D. & Pierantonio, A. (2020). *Supporting the understanding and comparison of low-code development platforms.* Paper presented at the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).
21. Sahinaslan, E., Sahinaslan, O. & Sabancıoglu, M. (2021). *Low-code application platform in meeting increasing software demands quickly: SetXRM.* Paper presented at the AIP Conf. Proc.
22. Salesforce. *Introducing Salesforce Platform Lightning*.
23. Sanchis, R., García-Perales, Ó., Fraile, F. & Poler, R. (2020). *Low-Code as Enabler of Digital Transformation in Manufacturing Industry*. Applied Sciences, 10(1), 12.
24. Sattar, N. A. (2018). *Selection of low-code platforms based on organization and application type.* Master's Thesis. Lappeenranta University of Technology School of Business and Management.
25. Shaikh, K. *Demystifying Azure AI*. Apress®.
26. Silva, C., Vieira, J., Campos, J. C., Couto, R. & Ribeiro, A. N. (2020). *Development and Validation of a Descriptive Cognitive Model for Predicting Usability Issues in a Low-Code Development Platform*. Human Factors, 0018720820920429.
27. Strømsted, R. I., Marquard, M. & Heuck, E. (2018). *Towards Low-Code Adaptive Case Management Solutions with Dynamic Condition Response Graphs, Subprocesses and Data.* Paper presented at the 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW).
28. Vikebø, E. & Sydvold, L. B. (2019). *An inquiry into low-code solutions in institutions for higher education: a case study of low-code implementation at the Admissions Office at the Norwegian School of Economics.*
29. Vincent, P., Iijima, K., Driver, M., Wong, J. & Natis, Y. (2019). *Magic quadrant for enterprise low-code application platforms*. Gartner report.
30. Vincent, P., Natis, Y., Iijima, K., Wong, J., Ray, S., Jain, A. & Leow, A. (2020). *Magic Quadrant for Enterprise Low-Code Application Platforms*. Retrieved from: <https://www.gartner.com/en/documents/3991199/magic-quadrant-for-enterprise-low-code-application-platf>
31. Warren, N. (2018). *Low-Code Myths, Fears, and Realities: Scalability.* Retrieved from https://www.outsystems.com/blog/posts/low-code-myths-scalability/.
32. Waszkowski, R. (2019). *Low-code platform for automating business processes in manufacturing*. IFAC-PapersOnLine, 52(10), 376-381.
33. Woo, M. (2020). *The rise of no/low code software development - No experience needed?* Engineering (Beijing, China).

* * *

Hana A. ALSAADI is a specialist in computer information systems. She graduated from the Faculty of Computing and Information Technology at King Abdulaziz University with a bachelor's degree in Information systems: Decision Support Systems -with second honors- in 2020. She started studying for her master's degree in Information Systems at the Faculty of Computing and Information Technology at King Abdulaziz University in 2020. She is interested in cyber-security, data science, and machine learning. She always tries to improve her knowledge and skills in different fields.

* * *

Dhefaf T. RADAIN is an IT technician in the Technical Affairs Unit at the Faculty of Computing and Information Technology, King Abdulaziz University with over 10 years of experience in the field. She graduated from the Faculty of Computer and Information Sciences in King Saud university with a bachelor degree in Information Technology with second honors degree in 2007. She started her master degree in Information System in 2020 at the Faculty of Computing and Information Technology, King Abdulaziz University. Her interest is in Networking, Security and E-Systems. She is planning to use her knowledge and experience in the development of herself personally and professionally. Her job has a lot of potentials for development and she is aiming to be one of the people who leave a print in the place.

* * *

Maysoon M. ALZAHRANI graduated in 2020 with a Bachelor of Information systems and Technology, from the College of Computer Science and Engineering at the University of Currently studying a Master's of Computer Information System at King Abdulaziz University Jeddah, Jeddah, Saudi Arabia.

* * *

Wahj F. ALSHAMMARI is a specialist in computer science. She graduated in 2019 with a bachelor's degree with honors in computer science from the College of Computers and Information Technology at the University of Tabuk. She started studying for her master's degree in information systems at the College of Computing and Information Technology at King Abdulaziz University in 2020. Interested in artificial intelligence, self-improvement, and art.

* * *

Dimah ALAHMADI received the B.Sc. degree in computer science and she holds an MBA degree from King Abdulaziz University (KAU), Saudi Arabia, and a Ph.D. degree in AI from the University of Manchester, U.K. Currently, she is working in the department of information systems, faculty of computing and information technology, KAU University. She is an assistant professor at KAU from 2016, where she is the supervisor of the Information system department. The area of her research interest is in AI, machine learning and data mining, decision support systems and Natural Language Processing.

* * *

Bahjat FAKIEH is working as assistant professor at King Abdulaziz University in Jeddah - Saudi Arabia. Previously, Bahjat was working as a lecturer in Federation University in association with ATMC college campus in Sydney for 1.5 years to teach cloud computing for a postgraduate program. In addition, he was working as an Information Systems tutor in Macquarie University for 4.5 years. His research interest is in technology adoption, information systems, electronic business, electronic commerce, IT Project management and cloud computing.