5-11-2023

UNDERSTANDING THE SOCIO-TECHNICAL ASPECTS OF LOW-CODE ADOPTION FOR SOFTWARE DEVELOPMENT

Syed Asad Ali Naqvi
Leuphana University of Lüneburg, nedianmm31@gmail.com

Markus Philipp Zimmer
Leuphana University of Lüneburg, maphzi@icloud.com

Rehan Syed
Queensland University of Technology, r.syed@qut.edu.au

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

UNDERSTANDING THE SOCIO-TECHNICAL ASPECTS OF LOW-CODE ADOPTION FOR SOFTWARE DEVELOPMENT

Research Paper

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

Zimmer, Markus Philipp, Leuphana University of Lüneburg, Germany, markus.zimmer@leuphana.de

Rehan, Syed, Queensland University of Technology, Australia, r.syed@qut.edu.au

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

Abstract

Organizations leverage several approaches for creating software applications that meet the requirements of specific contexts. Besides well-researched approaches like software development, outsourcing, or customizing commercial software packages, low-code platforms today offer a new approach for creating software. The low-code approach allows to develop software without or with limited actual coding by combining executable software components into workflows. While the low-code approach simplifies software development and reduces effort and time, we lack explanations on why organizations adopt it, and which challenges are associated with its adoption. We, therefore, investigate the adoption of the low-code approach based on the technology-organization-environment framework. We identified ten aspects supporting and six aspects hindering the adoption of the low-code approach. For practice, we propose a model that can assist organizations in determining the adequacy for adopting the low-code approach.

Keywords: Low-code approach, Low-code platforms, Technology adoption, Technology–Organization–Environment framework, Software development strategies.

1 Introduction

Digital transformation (DT) is vital to organizations' value creation, competitiveness and success (Vial 2019). The DT of organizations' leverages several approaches for creating software applications that meet the requirements of specific contexts (Sebastian et al. 2017). Besides well-researched and sophisticated approaches such as, outsourcing software development, or customizing commercial software packages, software development projects still have high failure rates (Engelbrecht et al. 2017; Zaman et al. 2019). The low-code approach offers a new approach for creating software by enabling software development based on Graphical User Interface (GUI) instead of traditional computer programming. According to Richardson and Rymer (2014), low-code is an enabler for accelerating customer-centric software development. Low-code offers software development via drag-and-drop of executable software components to form workflows. While low-code involves interfaces for performing programming activities such as code modifications, the often interchangeably used term no-code refers to software development without writing any code (Bhattacharyya and Kumar 2021). Low-code can therefore support a broader range of enterprise application development than no-code (Sahay et al. 2020). Since the use of the low-code approach in software development is still emerging, many organizations are indecisive about it's adoption to simplify their software development processes (Sahay et al. 2020).

Adoption refers to the diffusion and use of technology to gain productivity or economic benefits (Karahanna et al. 1999). We define low-code adoption as a technology and method of software development where both elements are intertwined. With the low-code approach, we study how the intertwining of the two elements, technology and method, can influence the adoption. Due to the widespread popularity of the low-code approach, many software providers today identify their products as low-code including sales automation, enterprise application development, business intelligence, machine learning, E-commerce, cybersecurity, and customer relationship management (Sahay et al. 2020; Richardson and Rymer 2014). In this study, we focus on low-code for enterprise application development. Low-code software development is enabled by low-code platforms (LCPs). Hence, the adoption of the low-code approach requires organizations to adopt an LCP. Thereby, onwards, we use the term low-code approach as the key concept and when referring to specific platforms, we use the term LCPs.

Krejci et al. (2021) investigated how the low-code approach facilitates the idea management process by empowering non-IT professionals. Indeed, low-code development enables non-IT professionals to take part in the software development process. Non-IT professionals without formal software development training who use low-code approaches for enterprise applications are called citizen developers (Gartner n.d.). Subsequently, citizen development is the phenomenon of engaging and empowering non-IT professionals in software development using the low-code approach. Citizen development has been traditionally argued to overcome challenges in software development, such as a shortage of developers (Duncan et al. 2021). For example, Krejci et al. (2021) showed that low-code helped organizations promote innovative behavior and autonomy, among their ordinary employees. In contrast, Hoogsteen and Borgman (2022) identify some risks associated with citizen development, including robust foundations, application security, and maintenance.

As there is no standard definition of low-code, we consider low-code adoption neither a standard packaged software solution (such as CRM) nor a new software development methodology (such as Scrum). Instead, low-code adoption is about adopting technology and, at the same time, adopting a new way of software development. Thus, the adoption of low code differs from existing software development approaches discussed in the literature. Despite the low-code approach's increasing popularity among practitioners, we lack empirical explanations why organizations opt for low-code as a software development strategy and which aspects hinder the adoption (Sahay et al. 2020). In addition, researchers have investigated low-code adoption only in technological contexts (Di Ruscio et al. 2022; Bock and Frank 2021). So far, research lacks an understanding of how a combination of technological, organizational, and environmental aspects impact organizations' decision for adopting the low-code as a software development approach (Iho et al. 2021). A better understanding could help to explain the aspects for the rapid growth of adoption as well as aspects hindering the adoption. Due to the high relevance for practice and a lack of research from a socio-technical perspective, we investigated the following research question: ***Which aspects influence the organizations to adopt the low-code approach for software development?***

To address this research question, we take an explorative approach to identify aspects that support or hinder the adoption of the low-code approach. Since the low-code approach brings together existing technologies, rather than offering new technologies in silos, such as new programming language, the low-code adoption is not purely technology adoption. Moreover, it enables new forms of organizing software development and can support organizations in tackling environmental shifts. Therefore, we selected TOE as the theoretical framework to have a broad perspective on how software development practices change due to low-code adoption. We decided against a deductive approach based on existing adoption models in this study to account for the newness of the technology and to preserve the openness in identifying relevant factors. In our study, we draw on 18 semi-structured expert interviews, as well as on archival data including the platform documentation and 654 user reviews of the three LCPs namely Pega, Appian and Outsystems. We identified ten aspects supporting and six hindering low-code adoption. The findings of this study enhance the understanding of the organizations' decision to adopt the low-code approach. Our contributions are two-fold. First, we contribute the field of software development approaches employed for DT. Second, by investigating the aspects that support or hinder the low-code approach adoption, we contribute to the nascent research stream of the low-code approach by developing a model that explains why organizations adopt low-code. This model can also be used to decide whether or not to adopt low-code as a strategy for software development.

This study continues with the following sections: Section 2 analyzes the current body of knowledge related to the challenges in software development, with a brief overview of the literature. We discuss the TOE framework as the theoretical lens employed in this study. Data collection and analysis methods are explained in Section 3. Section 4 presents a summary of key findings and a model for organizations to evaluate the low-code approach to software development. In section 5, we discuss the results and limitations of this study. Section 6 concludes the paper and suggests future research.

2 Existing work on software development and adoption

A large and growing body of literature has investigated software development approaches adopted by organizations including global software development (GSD), customization of packaged software, and model-driven engineering (MDE) (Herbsleb and James D. 2007; Jacobson and Bylund 2000; Brambilla et al. 2017). With the increasing demands of DT, many organizations have adopted packaged software, which consists of pre-built applications (CRM, ERP) that are customized to fit business needs. These applications require specialized skills and high maintenance with high upfront and maintenance costs (Light 2003; Holland and Light 1999). By exploiting offshore resources, Carmel (1999) found several organizations used GSD to overcome the domestic shortage of software developers, gain cost benefits, and reduce development time. However, many studies have identified the challenges that organizations face when using GSD (Mockus and Herbsleb; Prenner et al. 2021). Cultural differences, distance, and lack of trust have been identified as major factors affecting communication and collaboration between offshore and onshore teams, leading to poor software quality (Herbsleb and Moitra 2001; Smith and Ruiz 2020). Therefore, several organizations report that such strategies are inadequate to meet their software development needs (Herbsleb et al. 2001; Prenner et al. 2021).

MDE is a software development approach that divides the application into components that can be developed and tested independently, and later the modules can be merged (Brambilla et al. 2017). MDE facilitates flexibility and collaboration in distributed software development teams (Cusumano 2008). However, MDE is still considered a niche technology with several inadequacies, including lack of standardization in modeling libraries (Mussbacher et al. 2014; Di Ruscio et al. 2022). Furthermore, generic software development cannot capture the value of MDE, as opposed to domain-specific software development (Whittle et al. 2014). Di Ruscio et al. (2022) compared the similarities and differences between MDE and the low-code approach. They identified five categories that can characterize MDE and the low code approach. 1. MDE: Category one is exclusive to MDE. Such approaches do not aim to reduce coding efforts rather used for task automation. 2 & 3. MDE + Low-code application platforms: Categories two and three share similarities between MDE and low-code. Both categories use MDE and aim to reduce coding efforts for software development. However, category two lacks deployment or lifecycle management capabilities support, while category three offers it. 4 & 5. Low-code application platforms + Low-code software development: Categories four and five are exclusive to low-code and do not use model-driven approaches. Category four supports deployment or lifecycle management capabilities, while category five does not offer such support. Our study examines low-code approaches that use MDE for enterprise application development and offer end-to-end deployment or life cycle management capabilities, stemming from category three of Di Ruscio et al. (2022) analysis.

In contrast to existing software development strategies, the low-code approach enables software development using drag and drop via a GUI instead of writing software code. There are certain technological characteristics of the low-code approach discussed in the literature. Sahay et al. (2020) compared eight LCPs and identified nine technical features of the low-code approach that simplify software development. There are six features identified as mandatory, including GUI, interoperability, security, business logic specifications, and application build mechanisms. Additional optional features include collaborative development support, reusability support, and scalability (Sahay et al. 2020). In contrast, Bock and Frank (2021) described common features like data structures and GUI designer, API connectivity to external data sources, and occasional features like traditional coding components. According to Iho et al. (2021), five characteristics of the low-code approach support knowledge integration between IT and business users. Centralized control, reusable development components, ease of mastery, a visual user interface, and real-time editing are among these. By leveraging such characteristics, low-code approaches facilitate digital innovation, leverage business knowledge, and allow organizations to develop a common view between business and IT during software development. The adoption of the low-code approach is not just about using the technological features, rather, organizations also need to consider the changes in their software development practices (Richardson and Rymer 2014). This suggests that the adoption of low-code involves technological, organizational, and environmental contexts.

2.1 The technology–organization–environment framework

The TOE framework explains how innovations and IT services are adopted (Baker 2012). Its key premise: a combination of factors situated in the TOE contexts can explain IT adoption (Tornatzky et al. 1990). During the past decades, researchers have used the TOE framework to study the adoption of various technologies in organizations including Electronic Data Interchange (Kuan and Chau 2001), E-Commerce (Liu and Arnett 2000), Enterprise systems including ERP, SCM, CRM and e- procurement (Ramdani et al. 2009; Cruz-Jesus et al. 2019; Pan and Jang 2008), E-business systems (Oliveira and Martins 2010; Zhu and Kraemer 2005), social media (Abed 2020; Abeysinghe and Alsobhi 2013) , and cloud computing (Low et al. 2011; Borgman et al. 2013). This model has been widely used in many studies to explain why organizations adopt novel technologies (Bosch-Rekveldt et al. 2011; Awa et al. 2017; Oliveira and Martins 2011; Erind 2015). We use the TOE framework as a theoretical lens to understand and explain low-code adoption in this study.

The technological context: According to Thong (1999), technological innovations have four characteristics: nature, complexity, motivation, and timing of innovation. Nature refers to whether the innovation is product or process related. Complexity defines whether the innovation is radical or incremental. Motivation captures the innovation driver, i.e., market pull or technology push. Timing means if innovation is planned or incidental (Thong 1999). Chian (2010) added a fifth feature to the source of innovation, i.e., internal vs. external. Analyzing the existing literature on the technology context, Awa et al. (2017) identified three critical technology factors. These include perceived simplicity, which is related to ease of use; perceived compatibility, which refers to the extent to which the technological innovations support existing infrastructure; and perceived value, which refers the benefits of the existing technology.

The organizational context: The organizational context describes a variety of characteristics such as organization size, management structure, and support, culture, financial cost, organizational readiness, technical competence, employees' interest in technology adoption, and the supplementary resources available internally (Baker 2012). Studies suggest organizational factors are crucial in assessing technology readiness and adoption (Abed 2020; Awa et al. 2017). Organizations with organic and decentralized structures are more likely to adopt new technologies since employees in such organizations have dynamic roles and practice lateral communication (Baker 2012). Innovation adoption may be difficult for small organizations, reports Erind (2015). Since such organizations are frequently constrained by financial constraints and resource scarcity.

The environmental context: According to Baker (2012), government regulations and external events can support or hinder technology adoption. According to Farndale et al. (2021), due to recent changes in macro-environments, organizations have difficulty sourcing high-skilled STEM talent. Further, a recent trend of deglobalization and regulations favoring in-house capabilities over outsourcing and offshore are negatively impacting high-tech companies' software development strategy (Farndale et al. 2021). These factors make hiring and retaining software developers challenging.

3 Methodology

Since the low-code approach adoption is an emergent phenomenon, we decided to take an exploratory approach (Yin 2009). We created a rich dataset by conducting semi-structured expert interviews and using archival data including analyst reports and user reviews of LCPs.

3.1 Data collection

For collecting data, we conducted eighteen semi-structured expert interviews between August 2021 and June 2022. Each interviewee has worked on enterprise application development projects with experience in both low-code and traditional software development. Due to stringent legal requirements, we focused on financial institutions adopting the low-code approach. Edwards et al. (2018) stated that, several European financial institutions have alleged to be non-compliant, due to, shortcomings in due diligence, violations, and inadequate control of money laundering practices. As a result, they have been fined by authorities for millions and even billions of dollars (Edwards et al. 2018). Due to the legal requirements, financial institutions seek to digitalize their processes by leveraging the low-code approach (among others). Our unit of analysis is the project, in which the low-code approach is used for developing enterprise applications. We started with a preliminary list of interviewees using our professional network. Initially, we interviewed senior management contacts (n=13) who have influenced the organizations' decision to adopt low-code or have supervised low-code projects. We used snowball sampling, asking interviewees to recommend other interview candidates with a proven track record in low-code projects (Biernacki and Waldorf 1981). We aimed to develop an understanding of low-code adoption throughout the organization. We also expected that only looking from senior management perspectives could incur bias in our results. To mitigate this bias, we included junior and mid-level roles (n=5) who have experience working with LCPs. This produced 18 interviewees (IP1-IP18). Our interviewees are from the financial services (n=8), IT consulting (n=6), Big four firms (n=4). Thus, they bring rich experience of working on different low-code projects. We conducted the interviews, which lasted on average 55 minutes, online. All participants were assured of their anonymity and consented to the recording and transcription of interviews. Our interview questions were adjusted to reflect the interviewee's experience and competence, i.e., by focusing on the technological or organizational aspects of low-code software development. While interviewing, we considered the threats to validity including researcher bias and reactivity, as described by Maxwell (2012). We ensured that the interviewees could speak openly and are allowed to explain their answers in detail without any conflicts.

We anticipated interviewee bias, meaning that they might only highlight the positive aspects of adopting a low-code approach. To mitigate this bias, we collected user reviews from Trustradius[1] on LCPs. Trustradius is a widely used platform for business technology reviews. When collecting reviews, we chose three platforms, namely, Appian, Pegasystems, and Outsystems since these are the most common LCPs for enterprise application development and categorized as leaders and visionaries in Gartner's and Forrester's analyst reports (Vincent et al. 2020; Bratincevic and Koplowitz 2021). Platforms not widely used, or not known for enterprise application development, or not the core product of platform providers were deliberately excluded. As recommended by Myers and Newman's (2007), this archival data enabled us to triangulate evidence and investigate the phenomena of the adoption of the low-code approach from diverse perspectives. We collected 654 user reviews for the three LCPs.

3.2 Data Analysis

We first transcribed the interviews and conducted an uncommitted review. After that, we added archival data and user reviews. We followed grounded theory process, as proposed by Charmaz (2008). We started with open coding. During analysis, we let the codes emerge inductively (Fernández 2004). We assigned the statements to codes. For example, "involve more people into software development", "making IT-Business alignment easier", "a way to engage more business users" were coded as citizen development. We generated 424 codes. "Fast application development", "easy customization", "framework applications", "citizen development", "low risk", "in-house capabilities", "legal requirements" and "small development teams" were key themes supporting low-code adoption, whereas "barrier to creativity", "programmer's nostalgia", "technical jargon", "lack of ownership", and "vendor lock-in" were the themes hindering low-code adoption. We followed the integrated delayed literature review approach in the grounded theory proposed by Urquhart (2022). We checked our data against the existing literature on low-code. We conducted three iterations followed between data and literature until no additional themes emerged. Our emergent theory was then linked to theoretical frameworks based on the data. We found the TOE framework by Tornatzky et al. (1990) informs our theory since it examines innovation adoption from a technological, organizational, and environmental perspectives.

[1] www.trustradius.com

The low-code approach accounted for 240 of the 424 initial codes. Some codes were not directly referring to the low-code approach, rather were related to the challenges in traditional software development, including "deglobalization", "software quality" "customization of packaged solutions" and "organization culture and openness to new technologies". These codes were yet relevant to explain the low-code adoption in a broader context. Combining constant comparisons, we produced 52 themes by merging the codes. These themes were then abstracted into 16 first-order and 5 second-order concepts. Throughout the analysis, we employed coder's corroboration (Saldaña 2021), to ensure reliability. After discussing the discrepancies related to the coding, we refined the codes. The iterative coding procedure came to an end in the third round, when no significant discrepancies were observed.

4 Findings

We identified ten aspects driving the low-code adoption and six that hinder it. We present these aspects within the TOE framework. To identify the aspects supporting or hindering the low-code approach, we use labels consisting of the three dimensions of the TOE framework (T – technology, O – organization, E – environment) and the letters S (supporting) and H (hindering), i.e. TS illustrates technological aspects supporting the low-code approach adoption, while TH shows technological aspects hindering the low-code approach adoption.

4.1 The technology context in adopting the low-code approach

Within the technology context, we identified five aspects stemming from the interview coding that explain why organizations adopt the low-code approach for software developments and two perceived challenges associated with the low-code approach. These include (TS1) customization of packaged applications, (TS2) low enterprise system integrity risk, (TS3) framework application, (TS4) frequent application deployment, (TS5) scalability, (TH1) barrier to creativity, and (TH2) vendor lock-in.

TS1 Customization of packaged applications: The participants highlighted concerns when using packaged solutions such as ERP and CRM as an alternative to traditional software development. *"If packaged software can meet the complete end-to-end organizational needs, then most organizations would prefer to use that instead of developing applications from scratch. But what happens, is that organizations realize quickly that packaged software has many limitations and are difficult to customize" (IP 3).* Many organizations have existing applications, that are difficult to customize and maintain. However, discontinuing these applications may incur setbacks for the business. The low-code approach can be used to develop the standalone application as well as integrate it into existing applications. *"The low-code approach is flexible to act as an intermediate solution, which can be built very fast and can interact with the existing ERP and CRM solutions and other technologies" (IP 1).* It offers capabilities including code reusability ('code' is referred to as application functionality) and integration to ensure that the software development is faster, and the applications are easily scalable. *"Organizations can take advantage of a lot of technology by integrating with existing applications using low-code" (IP 4).*

TS2 Low enterprise system integrity risk: Enterprise system integrity risk is referred to as the risk that can cause an enterprise application to malfunction, due to a minor technical error in the code on a sublevel. Since in the low-code approach, enterprise code and experiment code are kept separate, software developers perceive a low enterprise system integrity risk. *"Although citizen developers can work independently and fix any errors in the actual application themselves, using the low-code approach the application is segregated in layers/modules, and thus, the changes do not negatively affect the overall system instantly" (IP 9).*

TS3 Framework application: With the emergence of framework applications developed following the low-code approach, many organizations can use such applications as 'pay-per-use' solutions that can help to meet their increasing DT needs without high up-front costs for set-up, infrastructure and large number of software developers. For instance, processes such as the onboarding of clients follow standard steps across the financial industry. In the context of the finance industry, Know-Your-Customer (KYC) is a standard term and is defined by Hodgson (2002) as "It should be the responsibility of the institution to become familiar with the financial affairs of their client to prevent them from mis-selling them inappropriate policies or investments*"*. Many financial institutions today use such framework applications that are built using the low-code approach and can be easily customized to fit their organizational identity. *"With pre-built solutions such as KYC, we have a ready-to-use application within a very short time" (IP 7).*

TS4 Frequent application deployment: In traditional software development, a strong emphasis is on frequent deployment to ensure software quality. Since it is easier to identify and resolve bugs after each deployment. However, frequent deployments cause significant downtime and high testing and maintenance costs, limiting scalability. As MDE is embedded in the low-code approach, the functionality of each module can be tested separately. Therefore, the deployments are less frequent. Moreover, the module can be reused and customized. *"The low-code approach follows MDE. As a result, scaling and maintaining these* applications *is much easier" (IP 12).*

TS5 Scalability: The participants reported skepticism in the early stages of the low-code approach adoption to support enterprise-level software development. *"At the beginning, we were skeptical about using the low-code approach. Our main reservations were whether it can support application development that is used at a large scale (across the organization). After looking at the use cases and market reports, we decided to give it a chance. We realized that it is easier to develop, scale, and maintain applications because of its standardized modules" (IP 7).*

TH1 Barrier to creativity: While the advantages of the low-code approach are recognized, during the interviews, we also noted some drawbacks. One challenge in the adoption of the low-code approach from technical perspectives is that software developers may associate the low-code approach as a barrier to creativity and effective use of their technical potential. At the project level, this could mean that the low-code approach isn't the best solution when there is a creative solution required because it is difficult to accomplish with drag and drop. Therefore, there can be resistance from developers and some developers may prefer not to work completely on low-code projects and show more interest in software development using computer programming. *"The low-code approach restricts our ability to customize and enforce several technical restrictions. It is like, even if you are fully capable of walking, would you prefer to use handicap equipment?" (IP 11).* The participant supported that the low-code approach follows a standardized software development approach, which in the beginning may seem inflexible. *"Low-code approach uses standardized modules, that take away a bit of flexibility but the return of it is that once you know a bit how it works then software development is much easier" (IP 4).* Another shortcoming of the low-code approach is whereas, the standard requirements can be fulfilled easily, customized requirements can be difficult to implement and often involve programming as the participant reported. *"If we need to do some customization then we need to write custom java scripts to achieve the functionality. Unfortunately, when upgrading such customization are not supported by the LCP" (IP 17). "LCP providers do not support faults that occur in customized solutions" (IP 15).*

TH2 Vendor lock-in: Vendor Lock-in is a term used in the IT sector to describe the dependency on a single platform provider. Similar to any commercial technology, the participants recognize the risks of vendor lock-in, when using the low-code approach. The participant also suggested being aware of the drawbacks of vendor lock-in, however, since the value proposition and technical features of most LCP providers are similar, they perceive vendor lock-in as a low-risk in the adoption of the low-code approach. *"Of course, switching between LCP providers is not easy. However, if you look at the most widely used low-code platforms, they are not much different in terms of technical features. Even if an LCP provider introduces a unique feature, because of the competition between the LCP providers, you can get it with the next release"* (IP 16). *"If you are using a platform, that offers you cutting-edge features for application development and is providing value and competitive advantage to your organization, then I consider that being locked-in with a vendor isn't that bad"* (IP 18).

4.2 The organizational context in adopting the low-code approach

In the organizational context, we found three aspects supporting the adoption and four aspects hindering the adoption of low-code approaches. The supporting aspects are (OS1) increased demand for digital applications, (OS2) enabling agility in software development, (OS3) enabling citizen development, (OH1) organization's culture and decision making, (OH2) programmer's nostalgia, and (OH3) shortage of platform experts, (OH4) technical jargon in the low-code learning.

OS1 Increased demands for digital applications: A clear benefit driving the low-code adoption in organizations is the increased demand for digital applications. The participant reported that organizations need to commit excessive resources for maintenance or customization of existing systems through programming.*"Every organization has a backlog that is growing because the demand for digital solutions is growing. Adding to that organizations need to commit large resources to the maintenance of existing systems. So, organizations faced with such challenges identify that something needs to be changed if they want to make software development faster" (IP 16).*

OS2 Enabling agility in software development: Organizations find it challenging to enable agility in software development. Since the list of requirements is usually incomplete and not fixed for enterprise applications, rather the requirements are developed and changed iteratively based on business needs and end-user feedback. The low-code approach supports agile software development by allowing experimentation at a low cost and enabling quick feedback. *"I think in situations where organizations are not 100% sure and need to explore, using the low-code approach for software development would be a better decision. Moreover, the team size is reduced drastically when using the low-code approach. Even one developer team exists on low-code projects" (IP 4). "Using the low-code approach, we have high development speed, and without any large development teams. We have software development projects that use the low-code approach and are just one developer team, and it works perfectly well. It can also be two or three people, however, I haven't seen a team of 10 developers working on a single low-code application" (IP 3).*

OS3 Enabling citizen development: Organizations can engage more workforce in software development and promote shared ownership between business users and software developers. Moreover, organizations can use the low-code approach as a strategy to overcome the shortage of software developers. *"Using low-code capabilities, citizen developers are empowered to make the changes in the application themselves. As we operate under competitive environments, I (technical lead) want to focus on business-critical topics and am happy if citizen developer experiment and does moderate changes to the application themselves" (IP 12).* Organizations perceive the low-code approach as a means to enable experimentation in software development at minimum costs. *"LCP enable us to prototype and test at minimal costs, thus making our businesses customer-centric and our operations agile" (IP 5).* We further investigated how the concept of citizen development can be implemented through the low-code approach and how organizations can ensure governance within citizen development. The participant emphasized establishing separate development environments for citizen developers. *"I would define a certain space where citizen developer would work, for example, in user interface. I think that's perfect because these smaller steps are easy to find in the low-code environment" (IP 3).* The changes to the application, complied by citizen developers are reviewed by the software developers and can be either rolled back or deployed to the enterprise level. This implies that citizen developers can do iterations and changes themselves, however, this experimentation does not break the enterprise application. Organizations can also improve task delegation using the low-code approach, as highlighted by the citizen developer *"The low-code approach also provide Rule-Delegation capabilities, where software developers can deliberately assign specific rules to citizen developers to manage seasonal fluctuations" (IP 8).*

OH1 Organization's culture and decision-making: The participant reported that among the organizational aspects that hinder the low-code approach adoption organizational culture and decision-making are key aspects. Organizations that do not promote empowerment within their employees and lack ownership may find it difficult to use the low-code approach. *"The low-code approach is not suitable for organizations that do not support fast decision-making or have top-down hierarchical decision-making processes" (IP 9).* Since the adoption of the low-code approach is more than introducing new technology, but also a new way of working, we also attempted to uncover how the organization's culture can support or hinder it. *"Organizations that are not agile may not like the fast* *application development process driven by the low-code approach" (IP 6). "I think the organizations where the low-code approach may fail, have a strong division between business and IT or if the vision for the end-user application is fixed" (IP 1).* Another participant reported that organizations may not find the low-code approach suitable, if they prefer to maintain long approval for change processes, application functionality or have a fixed vision for the end-user application, as *"If the approval processes take weeks or months then, in such cases, the organization would be better off with traditional programming" (IP 2).* This aspect of culture bears a risk that participants suggest that organizations can overcome by considering the initial two years as learning time. *"If organizations are trying the low-code approach for the first time, they should encourage experimentation and should expect some failures in the first two years. Based on the learnings, they should then adapt their software development strategy" (IP 12). "In the early phases of the low-code adoption, organizations should allocate resources to regular training for technical enablement. This will help to engage IT professionals and citizen developers and promote a culture of frequent feedback" (IP 5).* However, the participant also reported that due to this steady adoption of the low-code approach, it may not be best suited for small businesses. *"To execute the low-code approach, you need large upfront investments for license costs, training, and consulting. While the low-code technology is robust, I doubt that it is yet suitable to meet the small business needs" (IP13)*

OH2 Programmer's nostalgia: Programmer's nostalgia is another challenge that organizations may face, as some software developers may dislike the low-code approach because they prefer to code. We noted some reservations about the low-code approach. *"Contrary to our expectations, the first response of some software developers was instead of coding, I need to work on this? Or everyone (citizen developers) can do it now, so where is the value of my work? So, you need to convince them, that is quite difficult" (IP 12). "The low-code approach doesn't excite most of the developers as they normally love to code and find programming exciting. I think the LCP provider should help facilitating the transition for developers from programming to the low-code approach. It's a culture change not always easy to accomplish" (IP 13).*

OH3 Shortage of platform experts: The participants reported that there is a shortage of low-code experts and the low-code approach is often considered a niche skill because of the platform-specific expertise. Moreover, due to a shortage of low-code experts, organizations face a war of talent. Consequently, the low-code experts are paid higher than the traditional software developers with the same experience. *"Although we pay for low-code development roles significantly higher than the traditional programming roles, we still find it difficult to hire and retain LCP experts" (IP 7).*

OH4 Technical jargon in the low-code learning: Technical jargon is another challenge that hinders the adoption of the low-code approach, as each LCP provider uses different terminology for describing similar features. *"There is an urge between the LCP providers to market their features exclusively by using different terminologies. At its core, however, most LCP work the same and have similar features" (IP 15). "LCP requires a lot more setup and training. The certifications are also unique to each platform" (IP 18).*

4.3 The environmental context in adopting the low-code approach

The participants mentioned two aspects in the environmental context that support the low-code adoption (ES1) deglobalization and (ES2) stringent legal requirements in the finance industry.

ES1 Deglobalization: The interviewee reported that due to recent external events, organizations are seeking to move away from outsourcing software development, instead of focusing on building such capabilities in-house to gain better control over the application and deliverables. *"Due to the ongoing conflict in Ukraine, we are unable to operate from our offshore delivery centers. Furthermore, setting up an offshore delivery center elsewhere takes lots of time and resource investment. Thus, we are using the low-code approach to build and extend in-house software development capabilities" (IP 14).*

ES2 Stringent legal requirements: Due to stringent legal requirements in the financial sector, the participants reported banking and insurance organizations are the early adopters of the low-code. *"The finance industry has strict regulations and high costs for non-compliance. You would be a dissatisfied* *customer, if your banking application displays an incorrect balance or if your insurance provider is unable to locate your information" (IP 10).* Thus, due to the high cost of errors and regulatory and business consequences, many financial services organizations are using the low-code approach for software development instead of traditional programming. Moreover, since the processes, business requirements, and regulations are consistent across the financial industry, many LCP providers offer framework applications. The framework application is an off-the-shelf solution for standard processes such as client onboarding. Such applications require minimum customization, and thus, financial institutions can adhere to industry regulations within a very short time by using framework applications. *"I think that the benefit for using framework applications is to have the processes ready and just adapt the branding to their own company, through features such as user interface" (IP 8).*

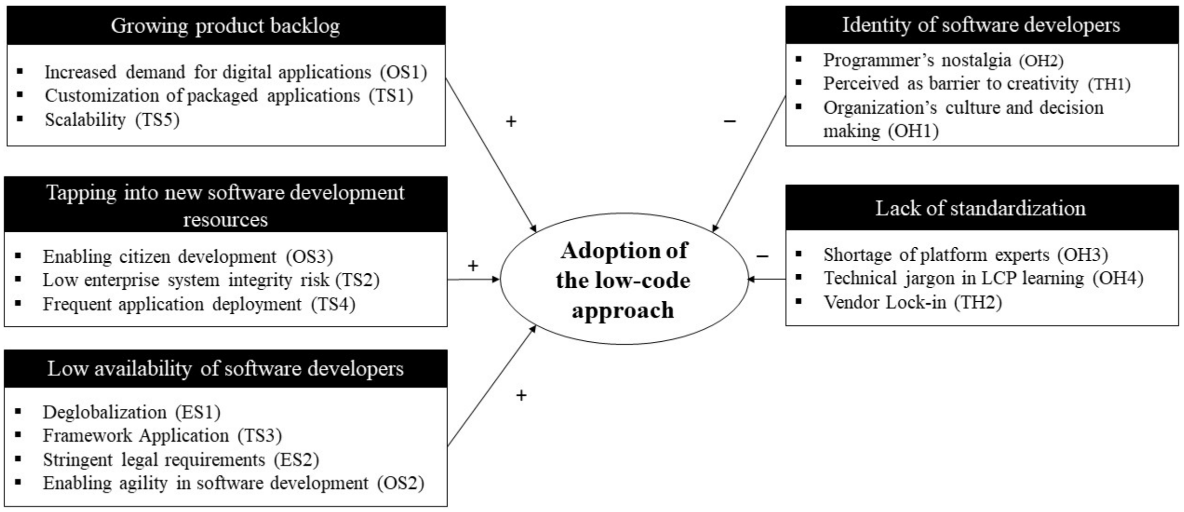
Growing product backlog
Increased demand for digital applications (OS1)
Customization of packaged applications (TS1)
Scalability (TS5)
Tapping into new software development resources
Enabling citizen development (OS3)
Low enterprise system integrity risk (TS2)
Frequent application deployment (TS4)
Low availability of software developers
Deglobalization (ES1)
Framework Application (TS3)
Stringent legal requirements (ES2)
Enabling agility in software development (OS2)
Identity of software developers
Programmer's nostalgia (OH2)
Perceived as barrier to creativity (TH1)
Organization's culture and decision making (OH1)
Lack of standardization
Shortage of platform experts (OH3)
Technical jargon in LCP learning (OH4)
Vendor Lock-in (TH2)
Adoption of the low-code approach
+
+
+
–
–

Figure 1 Aspects supporting and hindering the adoption of the low-code approach

5 Discussion

Organizations face several challenges in software development, including scarcity of software developers, high development and maintenance costs, and longer development cycles. To overcome these challenges, organizations have attempted various software development strategies including GSD, packaged solutions, and outsourcing software development. However, such strategies often fail to produce adequate software quality. Previous studies have discussed the shortcomings of such strategies, including lack of productivity and cultural differences resulting in poor software quality and high maintenance (Mockus and Herbsleb; Light 2003; Grundy et al. 2020). This study set out with the aim to assess the socio-technical aspects supporting or hindering the low-code adoption. To investigate the emerging phenomenon of the low-code adoption, we created a rich dataset by conducting 18 interviews and leveraging archival data including analyst report and user reviews. We used the TOE framework (Tornatzky et al. 1990) as the theoretical lens since it evaluates adoption from multiple contexts i.e., technological, organizational, and environmental contexts. By analyzing the results, we developed a model (see Figure 1) that illustrates the aspects supporting or hindering the low-code approach adoption. Since the success of the low-code adoption depends on the reasons the organization selects to use them in the first place, we identified three concepts supporting and two hindering the adoption. The concepts supporting include growing product backlog, tapping into new software development resources, and low availability of software developers, and the concepts hindering the low-code approach adoption are the identity of software developers, and lack of standardization in LCPs. In the following section, we discuss the implications of these findings.

The low-code approach promises to help organizations in meeting the increased demand for digital applications by tapping into new software development resources. Software applications built in-house instead of outsourcing are expected to be widely used within organizations, as the source of digital innovation influence its adoption (Chian 2010). Moreover, tapping into new resources for software development has become vital for organizations, as they face a severe scarcity of software developers locally, especially in developed countries, and find it challenging to hire and retain software developers (Farndale et al. 2021). In this study, we found that the low code approach helps organizations overcome the shortage of software developers in three ways. First, by using the low-code approach organizations can promote collaboration between software developers and non-IT professionals through citizen development. This makes it possible to include a wide range of employees in the software development process. This finding corroborates with Krejci et al. (2021), who investigated how organizations can leverage business-oriented LCPs to facilitate the radical innovation process. However, with our dataset, we focused on IT-oriented low-code approaches that can facilitate both radical and incremental innovations in organizations (Thong 1999), due to their technological characteristics for developing enterprise applications. Second, using the low-code capabilities including drag and drop interfaces, easy integration to external applications, and code reusability in software development, the team size is drastically reduced. This can help organizations overcome the shortage of software developers. These findings echo other studies on low-code adoption (Iho et al. 2021; Sahay et al. 2020; Bock and Frank 2021). Third, framework applications that are industry-specific ready-to-use applications are designed and updated using the low-code approach. Framework applications can be easily customized as per the organizational identity and can be used on a pay-per-use model. We discussed how framework applications are used by financial institutions to remain compliant against stringent regulations while reducing software development efforts. This finding is unique as such framework applications are offered by IT-oriented low-code approaches. We found that framework applications are currently very popular in financial services organizations.

We argue that the low-code approach can act as a middle way between software development from scratch and packaged solutions. In reviewing the literature, we identified that packaged solutions are unable to meet organizational demands for digital applications since such solutions are difficult to customize and have high implementation and maintenance costs (Light 2003). Thus, packaged solutions are difficult to scale and require high maintenance efforts. Whereas the strategy for developing digital applications from scratch through computer programming provides better control to the organization compared to packaged solutions. A major shortcoming of this approach is that the speed of development is slow and the risk of having an error in the application is high (Prenner et al. 2021; Mockus and Herbsleb). To minimize risks, when following this software development approach, the participants in this study reported using frequent deployment, testing, and shorter development cycles. However, the productivity of software development is severely affected when using such remedies. In this study, we found that the low-code approach can mitigate such risks in software development projects in two ways. First, organizations can bring flexibility to their software development projects, as the low-code approach supports faster feedback loops and can accommodate frequent change requests during software development. Thus, by using the low-code approach organizations can support iterative software development. Moreover, by enabling quick feedback and fast decision-making, the low-code approach bring agility to software development projects. Second, by adopting the low-code approach organizations can overcome the causality dilemma by involving both business and IT professionals simultaneously in the software development processes through citizen development. Prior studies have emphasized the IT-Business alignment for successful technology adoption (Venkatraman et al. 1993; Engelbrecht et al. 2017). Using the low-code approach capabilities such as task delegation, software developers enable business users to configure the options themselves, such as seasonal fluctuations in demand, supporting the changing business requirements. Due to such capabilities, the low-code approach promotes the active involvement of the business users and it seems to enable new forms of IT-business alignment in software development (Venkatraman et al. 1993; Gellweiler 2022; Engelbrecht et al. 2017). An implication of this finding changes completely how the backlog is organized and prioritized within organizations.

While some interviewees argued that the low-code approach is easier to use and can simplify software development by allowing for new ways to enable MDE (Di Ruscio et al. 2022), we also identified a number of issues related to the low-code approach. As reported in previous studies (Hoogsteen and Borgman 2022), customized software development requirements cannot be managed by default configurations of low-code but require programming. We extend the current research by uncovering the implications of this phenomenon. One unanticipated finding was that many LCP providers offer programming interfaces for customization. However, such customizations are not supported. This implies that while upgrading the application version, the customized features may need to be re-programmed, rendering the application maintenance laborious. This can hinder the adoption of the low-code approach, particularly for smaller organizations that have a scarcity of software developers and resources (Thong 1999; Erind 2015). Moreover, we identified that the setup and learning of the low-code approach can be challenging, due to the lack of standardization within LCPs. As each LCP provider intends to distinguish and present their offerings uniquely, despite similar technical capabilities, the low-code features are often described with technical jargon hindering the learning process. We compared the unique findings of our study to existing literature in table 1.

Existing research	**Our study's findings**
Growing product backlog	
Existing literature discusses low-code capabilities help in overcoming growing product backlog through a technical lens, comparing the existing low-code platform features (Sahay et al. 2020).	Low-code can help organizations tackle the challenge of growing product backlog through a combination of socio-technical aspects.
Tapping into new software development resources	
Prior literature acknowledges that IT-business alignment has several benefits. However, we lack explanation how such alignment can be achieved within the software development context (Krejci et al. 2021).	The low-code approach can lower the risks in software development by enabling citizen development, leveraging MDE principles and frequent application deployment.
Low availability of software developers	
Low-code is about enabling productivity in software development (Bock and Frank 2021). Using low-code, organizations can unlock the innovative potential of their employees (Krejci et al. 2021).	Low-code can help organizations overcome the developer shortage in severals ways, including framework applications that are pre-built for specific business processes. Moreover, low-code reduces the costs of experimentation and prototyping.
Identity of software developers	
Software developers find it easier to implement the requirements through programming in certain situations instead of leveraging the low-code features (Hoogsteen and Borgman 2022).	Due to software developers' nostalgia, they may prefer to code and perceive low-code as a barrier to creativity. Moreover, by comparing their roles to citizen developers, developers may feel their unique identity is endangered.
Lack of standardization	
For MDE, Eclipse provided a large and dynamic Ecosystem. Low-code also requires an ecosystem to support the continuing expansion of a sustainable user base (Di Ruscio et al. 2022).	We identified that lack of standardization and inconsistent use of terminology across the low-code platforms could be a severe challenge, hindering low-code adoption.

Table 1 Comparison of our study's findings on low-code adoption to existing research

We noted that the community systems are established by each platform provider and thus, such forums are controlled by LCP provider. In this study, we observed that such forums have a small community size, in comparison to open-source forums. Moreover, some participants have difficulties finding the right information on LCP forums because of the lack of standardized terminologies. This also contributes to the shortage of LCP experts as the knowledge exchange is currently limited across various LCPs. Thus, the low-code approach is yet considered a niche skill, with a small circle of users specific to LCP providers. An implication of this finding is that organizations find it difficult to hire and retain platform experts. In addition, depending on the certifications and expertise, such platform experts can be even more expensive than traditional programmers, a viable reason hindering the adoption of the low-code approach, particularly for small organizations. We also learned that organizations may find it challenging to promote the low-code approach among their existing software developers, who may perceive it as a barrier to creativity. Thus, some software developers may prefer to stay with traditional software development methods. A novel contribution of this study is that the low-code adoption may question the identity of software developers. The evidence from this study suggests that LCP providers should promote standardization and change management to overcome such barriers.

The findings of this study have certain limitations, most of which are grounded in the interviewees' sampling and can impact the theory's generalizability. First, we focused on the financial services sector to understand the low-code adoption. Though some of our interviewees have worked on other organization types, our primary focus remained on the finance sector, since we considered it as an early adopter of the low-code approach. Detailed focus on the organization types, e.g, retail, utility, or public sectors, and its effect on the adoption of the low-code approach can produce further interesting findings. Second, the interviewees' occupation is closely related to the use of LCPs, thus, they may exhibit bias towards the positive aspects of the low-code approach. We tackled this issue by explicitly asking them to compare the low-code approach to other software development approaches, they have used. Moreover, we used archival data, including user reviews to overcome this limitation. Third, the interviewees' are employed by global organizations in developed countries. The findings can be biased in favor of how the low-code adoption phenomena has developed there. A further study with focus on organizations' adoption of low-code from developing and emerging countries is therefore suggested.

6 Conclusion and outlook

Whereas, the low-code is increasingly recognized as a practical approach to accelerate DT, we lacked empirical explanations of the social-technical aspects that impact its adoption. This inspired us to investigate and uncover the aspects why organizations adopt the low-code approach for software development and the challenges associated with it. To analyze this emerging phenomenon, we created a rich dataset by conducting interviews and archival data. Using the TOE Framework as the theoretical lens, we investigated the relevance of the low-code approach and how it can help organizations overcome the challenges in software development. Based on the interview results, we developed a model that organizations can be used for evaluating the relevance of the low-code approach for software development. This research extends our knowledge of the low-code approach beyond technology factors and posits that despite the shortcomings of the low-code approach, it would become widely accepted as a viable foundation for accelerating software development. In this study, we found that by adopting the low-code approach, organizations can effectively gain control over their product backlog, reduce their reliance on software developers to create and maintain digital applications by tapping into new software development resources and overcome the shortage of software developers by leveraging the low-code capabilities including framework applications. We compared the low-code approach to existing software development strategies, and found that the low-code approach offer new forms to IT-business alignment, enables agility, and reduces risk in software development. On the other hand, the low-code approach may question the identity of software developers, since some programmers may have nostalgia to coding and perceive the low-code as barrier to their creativity. Moreover, technical jargon, and lack of support for customized solutions from LCP providers are the main obstacles in the adoption.

The low-code approach can contribute to creating sustainable digital futures. By transforming software development from a complex task that required high expertise to a team sport with a low barrier to technical expertise to participate, the low-code approach enables innovative ways of collaboration. Moreover, as the low-code approach reduces the resources required for developing digital applications, it makes software development, as a whole, more sustainable. The aspects driving the adoption of the low-code approach, identified in this study, illustrate that it is a promising approach to rapidly building enterprise applications. This study stimulates further research for the nascent stream of the low-code approach. We invite future studies to extend this model and identify the pathways that drive low-code adoption in organizations. It would also be interesting to contrast the results from this study with existing adoption models. Moreover, future research is needed to investigate how organizations can help developers manage the identity issues created by the low-code approach and incentivize both developers and non-IT professionals to adopt the low-code approach for software development.

References

Abed, Salma S. (2020): Social commerce adoption using TOE framework: An empirical investigation of Saudi Arabian SMEs. In *International Journal of Information Management* 53, pp. 102–118. DOI: 10.1016/j.ijinfomgt.2020.102118.

Abeysinghe, Geetha; Alsobhi, Aisha (2013): Social media readiness in small businesses. In *International Conference Information Systems*, Lisbon, Portugal. Available online at https://eprints.mdx.ac.uk/11433/1/social%20media%20readiness%20in%20s.

Awa, Hart O.; Ojiabo, Ojiabo Ukoha; Orokor, Longlife E. (2017): Integrated technology-organization-environment (T-O-E) taxonomies for technology adoption. In *Journal of Enterprise Information Management* 30 (6), pp. 893–921. DOI: 10.1108/JEIM-03-2016-0079.

Baker, Jeff (2012): The Technology–Organization–Environment Framework. In : Information Systems Theory: Springer, New York, NY, pp. 231–245.

Bhattacharyya, Som Sekhar; Kumar, Saurabh (2021): Study of deployment of "low code no code" applications toward improving digitization of supply chain management. In *JSTPM*. DOI: 10.1108/jstpm-06-2021-0084.

Biernacki, Patrick; Waldorf, Dan (1981): Snowball Sampling: Problems and Techniques of Chain Referral Sampling. In *Sociological Methods & Research* 10 (2), pp. 141–163. DOI: 10.1177/004912418101000205.

Bock, Alexander C.; Frank, U. (2021): Low-Code Platform. In *Bus. Inf. Syst. Eng.* 63, pp. 733–740. DOI: 10.1007/s12599-021-00726-8.

Borgman, Hans P.; Bahli, Bouchaib; Heier, Hauke; Schewski, Fiona (2013): Cloudrise: Exploring Cloud Computing Adoption and Governance with the TOE Framework. In *46th Hawaii international conference on system sciences* Wailea, HI, USA. DOI: 10.1109/hicss.2013.132.

Bosch-Rekveldt, Marian; Jongkind, Yuri; Mooi, Herman; Bakker, Hans; Verbraeck, Alexander (2011): Grasping project complexity in large engineering projects: The TOE (Technical, Organizational and Environmental) framework. In *International Journal of Project Management* 29 (6), pp. 728–739. DOI: 10.1016/j.ijproman.2010.07.008.

Brambilla, Marco; Cabot, Jordi; Wimmer, Manuel (2017): Model-Driven Software Engineering in Practice: Second Edition: Morgan & Claypool Publishers (3).

Bratincevic, John; Koplowitz, Rob (2021): The Forrester Wave™: Low-Code Development Platforms For Professional Developers, Q2. In *Analyst Report. Forrester Research Inc.* Available online at https://www.forrester.com/report/The-Forrester-Wave-LowCode-Development-Platforms-For-Professional-Developers-Q2-2021/RES161668, checked on 10/28/2022.

Carmel, Erran (1999): Global Software Teams: Collaborating Across Borders and Time Zones. In *Prentice Hall PTR*. DOI: 10.1080/15228053.2002.10855994.

Charmaz, Kathy (2008): Constructionism and the grounded theory method. Handbook of constructionist research (pp. 397–412). With assistance of J.A. Holstein & J.F. Gubrium (Eds.). New York, N.Y.: Guilford Press.

Chian, Felix Tan Ter. (2010): A perception-based model for technological innovation in small and medium enterprises. In *European Conference on Information Systems* Pretoria, South Africa. Available online at https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1011&context=ecis2010.

Cruz-Jesus, Frederico; Pinheiro, Andreia; Oliveira, Tiago (2019): Understanding CRM adoption stages: empirical analysis building on the TOE framework. In *Computers in Industry* 109, pp. 1–13. DOI: 10.1016/j.compind.2019.03.007.

Cusumano, Michael A. (2008): Managing software development in globally distributed teams. In *Communications of the ACM* 51 (2), pp. 15–17. DOI: 10.1145/1314215.1314218.

Di Ruscio, Davide; Kolovos, Dimitris; Lara, J. de; A. Pierantonio; M. Tisi; M. Wimmer (2022): Low-code development and model-driven engineering: Two sides of the same coin? In *Software and Systems Modeling* 21, pp. 437–446. DOI: 10.1007/s10270-021-00970-2.

Duncan, Ian; Jamnadass, Arjun; Magimay, Alwin (2021): Citizen Development. The Handbook for Creators and Change Makers. Chicago: Project Management Institute, checked on 9/15/2021.

Edwards, G.; Duran, C.; Sacchi, F.; Ackermann, B.; Conversano, L. (2018): Déjà Vu All Over Again: Money-Laundering And Sanctions Woes Continue To Haunt Europe's Banks. In *S&P Global Ratings 16*. Available online at https://www.allnews.ch/sites/default/files/ratingsdirect_dejavualloveragainmoneylaunderingandsanctionswoescontinuetohaunteuropesbanks_39968877_oct-16-2018.pdf, checked on 1/19/2022.

Engelbrecht, Jacus; Johnston, Kevin Allan; Hooper, Val (2017): The influence of business managers' IT competence on IT project success. In *International Journal of Project Management* 35 (6), pp. 994–1005. DOI: 10.1016/j.ijproman.2017.04.016.

Erind, Hoti (2015): The technological, organizational and environmental framework of IS innovation adaption in small and medium enterprises. In *International Journal of Business and Management* 3 (4), pp. 1–4. Available online at https://scholar.archive.org/work/trr7g57vtbbjxez4zzeafcniwq/access/wayback/http://www.iises.net/international-journal-of-business-management/publication-detail-255?download=1.

Farndale, Elaine; Thite, Mohan; Budhwar, Pawan; Kwon, Bora (2021): Deglobalization and talent sourcing: Cross-national evidence from high-tech firms. In *Human Resource Management* 60 (2), pp. 259–272. DOI: 10.1002/hrm.22038.

Fernández, Walter D. (2004): The grounded theory method and case study data in IS research: issues and design. In *Information Systems Foundations Workshop: Constructing and Criticising.* 1 (22).

Gartner (n.d.): Citizen Developer - Gartner IT Glossary, n.d. Available online at https://www.gartner.com/en/information-technology/glossary/citizen-developer, checked on 7/16/2021.

Gellweiler, Christof (2022): IT architects and IT-business alignment: a theoretical review. In *Procedia Computer Science* 196, pp. 13–20. DOI: 10.1016/j.procs.2021.11.067.

Grundy, John; Khalajzadeh, Hourieh; Mcintosh, Jennifer (2020): Towards Human-centric Model-driven Software Engineering. In : Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering: SCITEPRESS - Science and Technology Publications.

Herbsleb; James D. (2007): Global Software Engineering: The Future of Socio-technical Coordination. In : Future of Software Engineering (FOSE ’07): IEEE.

Herbsleb, J. D.; Mockus, A.; Finholt, T. A.; Grinter, R. E. (2001): An empirical study of global software development: distance and speed. In : Proceedings of the 23rd International Conference on Software Engineering: IEEE Comput. Soc.

Herbsleb, J. D.; Moitra, D. (2001): Global software development. In *IEEE Software* 18 (2), pp. 16–20. DOI: 10.1109/52.914732.

Hodgson, Damian (2002): “Know your customer”: marketing, governmentality and the “new consumer” of financial services. In *Management Decision* 40 (4), pp. 318–328. DOI: 10.1108/00251740210426312.

Holland, C. P.; Light, B. (1999): Global enterprise resource planning implementation. In *Hawaii International Conference on Systems Sciences* Maui, HI, USA. DOI: 10.1109/hicss.1999.772762.

Hoogsteen, Doortje; Borgman, Hans (2022): Empower the Workforce, Empower the Company? Citizen Development Adoption. In *Hawaii International Conference on System Sciences* Maui, HI, USA. DOI: 10.24251/hicss.2022.575.

Iho, Satu; Krejci, Désirée; Missonier, Stephanie (2021): Supporting Knowledge Integration with Low-Code Development Platforms. In *European Conference on Information Systems* Marrakech, Morocco. Available online at https://serval.unil.ch/resource/serval:bib_3ab938a18ba5.p001/ref.pdf.

Jacobson, Ivar; Bylund, Stefan (2000): The road to the unified software development process. Cambridge: Cambridge University Press, in association with SIGS Books.

Karahanna, Elena; Straub, Detmar W.; Chervany, Norman L. (1999): Information Technology Adoption Across Time: A Cross-Sectional Comparison of Pre-Adoption and Post-Adoption Beliefs. In *MIS Quarterly* 23 (2), p. 183. DOI: 10.2307/249751.

Krejci, Désirée; Iho, Satu; Missonier, Stéphanie (Eds.) (2021): Innovating with employees: an exploratory study of idea development on low-code development platforms. ECIS. Available online at https://serval.unil.ch/resource/serval:bib_fa487723156a.p001/ref.pdf.

Kuan, Kevin K.Y.; Chau, Patrick Y.K. (2001): A perception-based model for EDI adoption in small businesses using a technology–organization–environment framework. In *Information & Management* 38 (8), pp. 507–521. DOI: 10.1016/s0378-7206(01)00073-8.

Light, Ben (2003): CRM packaged software: a study of organisational experiences. In *Business Process Management Journal* 9 (5), pp. 603–616. DOI: 10.1108/14637150310496712.

Liu, Chang; Arnett, Kirk P. (2000): Exploring the factors associated with Web site success in the context of electronic commerce. In *Information & Management* 38 (1), pp. 23–33. DOI: 10.1016/s0378-7206(00)00049-5.

Low, Chinyao; Chen, Yahsueh; Wu, Mingchang (2011): Understanding the determinants of cloud computing adoption. In *Industrial Management & Data Systems* 111 (7), pp. 1006–1023. DOI: 10.1108/02635571111161262.

Maxwell, Joseph Alex (2012): Qualitative research design. An interactive approach / Joseph A. Maxwell. 3rd ed. Thousand Oaks: Sage Publications (Applied social research methods, v. 41).

Mockus, A.; Herbsleb, J.: Challenges of global software development. In *Proceedings seventh international software metrics symposium* IEEE, pp. 182–184. DOI: 10.1109/metric.2001.915526.

Mussbacher, Gunter; Amyot, Daniel; Breu, Ruth; Bruel, Jean-Michel; Cheng, Betty H. C.; Collet, Philippe et al. (2014): The Relevance of Model-Driven Engineering Thirty Years from Now. In *International Conference on Model Driven Engineering Languages and Systems* Valencia, Spain. DOI: 10.1007/978-3-319-11653-2_12.

Myers and Newman's (2007): The qualitative interview in IS research: Examining the craft. Information and organization. In *Information and Organization* 17 (1), pp. 2–26. Available online at https://www.pm.lth.se/fileadmin/_migrated/content_uploads/5._Qual_Interview_Texto_Leitura_Atividade_2.pdf.

Oliveira, Tiago; Martins, Maria F. (2010): Understanding e-business adoption across industries in European countries. In *Industrial Management & Data Systems* 110 (9), pp. 1337–1354. DOI: 10.1108/02635571011087428.

Oliveira, Tiago; Martins, Maria Fraga (2011): Literature Review of Information Technology Adoption Models at Firm Level. In *EJISE* 14 (1), pp110-121-pp110-121. Available online at https://academic-publishing.org/index.php/ejise/article/view/389.

Pan, Ming-Ju; Jang, Woan-Yuh (2008): Determinants of the Adoption of Enterprise Resource Planning within the Technology-Organization-Environment Framework: Taiwan's Communications Industry. In *Journal of Computer Information Systems* 48 (3), pp. 94–102.

Prenner, Nils; Unger-Windeler, Carolin; Schneider, Kurt (2021): Goals and challenges in hybrid software development approaches. In *Journal of Software: Evolution and Process* 33 (11), e2382. DOI: 10.1002/smr.2382.

Ramdani, Boumediene; Kawalek, Peter; Lorenzo, Oswaldo (2009): Predicting SMEs' adoption of enterprise systems. In *JEIM* 22 (1/2), pp. 10–24. DOI: 10.1108/17410390910922796.

Richardson, Clay; Rymer, John. (2014): New Development Platforms Emerge For Customer-Facing Applications (Forrester). Available online at https://www.forrester.com/report/New-Development-Platforms-Emerge-For-CustomerFacing-Applications/RES113411, checked on 7/16/2021.

Sahay, Apurvanand; Indamutsa, Arsene; Di Ruscio, Davide; Pierantonio, Alfonso (2020): Supporting the understanding and comparison of low-code development platforms. In *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE* Portorož, Slovenia. DOI: 10.1109/seaa51224.2020.00036.

Saldaña, Johnny (2021): The coding manual for qualitative researchers. Fourth edition. Los Angeles: SAGE (Core textbook).

Sebastian, Ina M.; Ross, Jeanne W.; Beath, Cynthia; Mocker, Martin; Moloney, Kate G.; Fonstad,.Nils O. (2017): How Big Old Companies Navigate Digital Transformation. In *MIS Quarterly Executive* 16 (3), pp. 133–150. DOI: 10.4324/9780429286797-6.

Smith, Sarah Morrison; Ruiz, Jaime (2020): Challenges and barriers in virtual teams: a literature review. In *SN Applied Sciences* 2 (6), pp. 1–33. DOI: 10.1007/s42452-020-2801-5.

Thong, James Y.L. (1999): An Integrated Model of Information Systems Adoption in Small Businesses. In *Journal of Management Information Systems* 15 (4), pp. 187–214. DOI: 10.1080/07421222.1999.11518227.

Tornatzky, Louis G.; Fleischer; Mitchell; Chakrabarti, Alok K. (1990): Processes of technological innovation: Lexington books. Available online at https://agris.fao.org/agris-search/search.do?recordid=us201300694725.

Urquhart, Cathy (2022): Grounded theory for qualitative research. A practical guide. Second edition. Thousand Oaks: Sage Publications.

Venkatraman, N.; Henderson, John C.; Oldach, Scott (1993): Continuous strategic alignment: Exploiting information technology capabilities for competitive success. In *European Management Journal* 11 (2), pp. 139–149. DOI: 10.1016/0263-2373(93)90037-i.

Vial, Gregory (2019): Understanding digital transformation : A review and a research agenda. In *The Journal of Strategic Information Systems* 28 (2), pp. 118–144. DOI: 10.4324/9781003008637-4.

Vincent, Paul; Natis, Yefim; Iijima, Kimihiko; Wong, Jason; Ray, Saikat; Jain, Akash; Leow, Adrian (2020): Gartner Magic Quadrant for Enterprise Low-Code Application Platforms. In *Analyst Report. Gartner, Inc.* Available online at https://www.gartner.com/en/documents/3991199, checked on 8/31/2021.

Whittle, Jon; Hutchinson, John; Rouncefield, Mark (2014): The State of Practice in Model-Driven Engineering. In *IEEE Software* 31 (3), pp. 79–85. DOI: 10.1109/ms.2013.65.

Yin, Robert K. (2009): Case Study Research: Design and Methods. In *The Canadian Journal of Action Research* 1 (14), pp. 69–71.

Zaman, Umer; Jabbar, Zulaikha; Nawaz, Shahid; Abbas, Mazhar (2019): Understanding the soft side of software projects: An empirical study on the interactive effects of social skills and political skills on complexity – performance relationship. In *International Journal of Project Management* 37 (3), pp. 444–460. DOI: 10.1016/j.ijproman.2019.01.015.

Zhu, Kevin; Kraemer, Kenneth L. (2005): Post-Adoption Variations in Usage and Value of E-Business by Organizations: Cross-Country Evidence from the Retail Industry. In *Information Systems Research* 16 (1), pp. 61–84. DOI: 10.1287/isre.1050.0045.