MIS Quarterly Executive

Volume 23 | Issue 3 Article 6

September 2024

Governing Citizen Development to Address Low-Code Platform Challenges

Altus Viljoen

Marija Radić

Andreas Hein

John Nguyen

Helmut Krcmar

MIS QUARTERLY EXECUTIVE

Governing Citizen Development to Address Low-Code Platform Challenges

Low-code development platforms empower citizen developers to develop software quickly and with minimal technical expertise. However, entrusting software development to novices risks substandard software quality, shadow IT and technical debt. Drawing from 30 interviews with citizen developers and low-code platform experts, we offer guidance for governing citizen development to address these challenges. We emphasize the crucial roles of technical experts and platform-specific functionalities in governing citizen development and demonstrate how citizen development governance diverges from conventional software development governance.[1,2]

Altus Viljoen
Technical University of Munich (Germany)

Marija Radić
Technical University of Munich (Germany)

Andreas Hein
University of St. Gallen (Switzerland)

John Nguyen
Technical University of Munich (Germany)

Helmut Krcmar
Technical University of Munich (Germany)

Strong Governance Is Needed to Mitigate Citizen Development Challenges

The shortage of skilled software application developers is widely acknowledged as a major constraint on digital transformation,[3] with IT leaders claiming that attracting and retaining skilled people has "never been more difficult than it is today."[4] In the U.S. alone, the shortage of software developers is expected to exceed 1.2 million by 2026.[5] Low-code development platforms such as Mendix and Microsoft Power Platform have emerged as tools to mitigate

1 Jonny Holmström is the senior accepting editor for this article.
2 The authors sincerely thank the senior editor and the members of the review team for their invaluable feedback throughout the review process.
3 Koehler, J. *The Biggest Obstacle to Digitalization*, T-Systems International GmbH blog post, February 14, 2023, available at https://www.t-systems.com/de/en/insights/newsroom/expert-blogs/how-to-beat-the-digital-skills-shortage-579652.
4 *IT Leaders Pulse Report 2022: Insights from 1,000 IT Leaders on People, Processes, and Technology*, MuleSoft, 2022, available at https://www.insightsforprofessionals.com/it/leadership/it-leaders-pulse-report-2022.
5 Zilberman, A and Ice, L. "Why Computer Occupations Are Behind Strong STEM Employment Growth in the 2019-29 Decade," *Beyond the Numbers* (10:1), U.S. Bureau of Labor Statistics, January 2021, available at https://www.bls.gov/opub/btn/volume-10/why-computer-occupations-are-behind-strong-stem-employment-growth.htm.

DOI: 10.17705/2msqe.00100

the skills shortage through "democratizing" software development. By offering drag-and-drop user interfaces and the ability to automatically generate code, these platforms remove the technical intricacies of software development to such an extent that workers without deep technical knowledge—so-called "citizen developers"—can contribute to application development efforts. Deploying low-code development platforms thus reduces the need for firms to recruit costly IT talent. The outlook for these platforms is bullish: Gartner predicts that 80% of applications will eventually be created by individuals outside of traditional IT roles.[6]

However, despite the advantages offered by low-code development platforms in expanding the pool of application developers, this approach introduces new challenges that organizations must navigate. First, the reliance on citizen developers raises concerns about the *quality of software* produced.[7] Second, *shadow IT* emerges as a significant risk. Citizen developers in various departments may independently create applications, making it difficult for the organization to monitor software efforts and ensure quality. The consequences of shadow IT vary, from serious issues like not complying with security and data protection standards, to less serious ones such as lacking consistency in corporate identity across applications.[8] Third, adopting these platforms can lead to the accumulation of *technical debt*, as applications developed by citizen developers may require extensive maintenance and optimization.[9] For example, new code branches added with a low-code development platform are often directly added to the code instead of being defined by a higher-level object class. This practice deviates from the standard software development approach of minimizing code redundancy, thereby complicating maintenance.

In this article, we offer guidance on governing citizen development to address these three challenges. First, we show that a more nuanced role categorization that goes beyond the "binary" distinction between "citizen developers" and "technical experts" informs robust citizen development governance. Second, we provide three sets of governance recommendations, with three specific recommendations in each set on how to strategically design, support and control citizen development, especially considering the collaboration between citizen developers and technical experts. Third, we show how governing citizen development differs from traditional software development and thus requires different governance mechanisms. Overall, our study shows that democratized application development must be underpinned by strong governance if citizen development is to become a viable complementor to firms' software development efforts.

Before presenting the findings of our study, we first provide a brief description of low-code development platforms and citizen developers, explain the focus of the study, describe the study design and case study partners and describe the three challenges in more detail.

Description of Low-Code Development Platforms and Citizen Developers

Low-code development platforms[10] are software tools that enable users to create applications with minimal hand coding, typically using visual configuration and interfaces rather than traditional programming languages. These platforms provide a range of features, such as drag-and-drop interfaces, prebuilt components and templates, allowing users to quickly develop and deploy custom applications without requiring coding expertise. They are designed to streamline the software development process, increase

6 *Gartner Says the Majority of Technology Products and Services Will Be Built by Professionals Outside of IT by 2024*, Gartner, Inc. press release, June 14, 2021, available at https://www.gartner.com/en/newsroom/press-releases/2021-06-10-gartner-says-the-majority-of-technology-products-and-services-will-be-built-by-professionals-outside-of-it-by-2024.

7 Lethbridge, T. C. "Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering," *Proceedings of 10th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2021*, Rhodes, Greece, October 17-29, 2021, pp. 202-212.

8 Elshan, E., Dickhaut, E. and Ebel, P. "An Investigation of Why Low Code Platforms Provide Answers and New Challenges," *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, Hawaii, January 2023.

9 Gartner, Inc. press release, op. cit., June 14, 2021.

10 Richardson, C. and Rymer, J. R. *New Development Platforms Emerge for Customer-Facing Applications*, Forrester, June 9, 2014, available at https://www.forrester.com/report/New-Development-Platforms-Emerge-For-CustomerFacing-Applications/RES113411.

productivity and empower users to create applications more efficiently.

Though all low-code development platforms aim to simplify software development, there is a wide diversity of platforms intended for different use cases and users. These platforms can be distinguished by their scope, complexity and target users. First, in terms of scope, some low-code development platforms are designed to be "general purpose"—i.e., they support a wide range of application development needs across various industries and use cases. Examples of such platforms include Mendix and OutSystems, both of which can be used to develop various mobile and web apps. The scope of other platforms is narrower, with a focus on specific industries or use cases. A popular example is workflow automation or robotic process automation (RPA) tools that allow users to optimize their processes, such as those offered by Kissflow, ServiceNow and Celonis.

Second, low-code development platforms differ in their complexity, specifically the degree of technical knowledge required to build applications on the platform. Mendix, for example, significantly reduces technical complexity but still requires users to have some basic programming knowledge to fully customize its out-of-the-box offerings. Other platforms, however, such as Bubble and Airtable, aim to remove coding completely and are referred to as "no-code" platforms.

Third, a platform's scope and complexity determine its target users. For example, because Mendix and OutSystems are aimed at developing fully fledged applications that still require some coding knowledge, they are less suitable for technical novices. Other platforms such as Microsoft Power Automate are aimed at specific use cases with low complexity and are more suitable for users without any programming knowledge.

Thus, given the wide variety of low-code development platforms, it is challenging to precisely define the profile of a citizen developer. Nevertheless, citizen developers have two consistent attributes: 1) they do not possess formal software development training; and 2) they typically work in business units and develop applications for those units. Gartner defines a citizen developer as "an employee who creates application capabilities for consumption by themselves or others" and notes that "a citizen developer is a persona, not a title or targeted role [and] report to a business unit or function other than IT."[11]

These attributes and definitions show that citizen developers do not develop complex, comprehensive software systems but instead work on smaller applications in their own business units that can address the practical problems they face. Good examples are an app developed by an American Red Cross employee that digitized paper-based supply chain processes to reduce order lifecycles[12] and an app developed by an electricity utility's employee during the COVID-19 pandemic to assist with access control to offices.[13]

Focus of Our Research Study

The goal of our research study was to investigate how firms can govern citizen development to mitigate the three challenges arising from citizen development: software quality, shadow IT and technical debt. To begin, we reviewed the extant literature on citizen development and identified four prevailing focus areas in the literature for addressing the challenges. Nevertheless, despite valuable insights from each focus area, they provide insufficient guidance for addressing our research goal. Table 1 outlines these focus areas and highlights the knowledge gaps and their implications for our study.

Our study addresses these four knowledge gaps by focusing on organizational, governance-related measures for how technical experts can assist citizen developers in mitigating the challenges associated with low-code development platforms.

11 *Citizen Developer*, Gartner Glossary, available at https://www.gartner.com/en/information-technology/glossary/citizen-developer.

12 Bhangar, S. *American Red Cross: Power Platform Customer Success Story*, Microsoft Power Apps Blog, August 1, 2018, available at https://www.microsoft.com/en-us/power-platform/blog/power-apps/americanredcross/.

13 *MBAS 2021: Low-code Developers at Eskom Create Apps for Building Access, Fleet Management, and More*, Microsoft Power Platform, May 2021, available at https://www.youtube.com/watch?v=r-IFkT3xKPc.

Table 1: Focus Areas of Extant Literature, Knowledge Gaps and Study Implications

Focus of Extant Studies	Description and Knowledge Gap	Research Need and Opportunity	Implications for Study Focus
Focus on Technical Solutions	Studies have predominantly examined low-code platforms from a technical perspective[14] or addressed technical problems at a feature level.[15] However, there is limited insight into how the "people-related" and organizational challenges of these platforms can be addressed.[16]	Though some studies have addressed organizational phenomena and challenges of low-code platforms,[17,18] there remain gaps in the understanding of organizational challenges caused by the "people aspect" of citizen development.	Our study addresses this gap by focusing on citizen development governance. We show that organizational factors can play a significant role in mitigating substandard software quality, shadow IT and technical debt.
Narrow Scope and "Single-User" Focus	Multiple studies have examined individual-level aspects of low-code development platforms.[19,20] Though such studies provide valuable insights, they do not consider the broader challenges of implementing low-code platforms or the different user types affected.	Broader, organization-wide challenges such as shadow IT and technical debt have been mentioned in extant literature.[21] However, there is still a lack of understanding in addressing these broader issues.	Our study looks beyond "single" citizen developers and explores how different actors collaborate during citizen development processes. We show that collaborative citizen development can mitigate some of the issues introduced by citizen development.
Experts' Involvement in Addressing Citizen Developers' Limited Knowledge	Though there are examples of citizen developers reaching out to experts when they have questions, these interactions have been studied only in forum-based contexts (e.g., asking questions on StackOverflow).[22] The questions relate purely to feature-level issues of low-code platforms.	Public online forums can assist citizen developers with technical problems but cannot address challenges unique to a company. In-house experts can provide more tailored support, focusing on a company's specific needs.	In line with the "multi-user" focus noted above, our study specifically investigates the role that technical experts in organizations play in assisting citizen developers in upholding high software quality standards. We show that experts play a vital role in doing so.

14 Ibid.

15 Bucaioni, A., Cicchetti A. and Ciccozzi, F. "Modelling in Low-Code Development: A Multi-Vocal Systematic Review," Software and Systems Modeling (21), January 2022, pp. 1959-1981.

16 Prinz, N., Rentrop, C. and Huber, M. "Low-Code Development Platforms—A Literature Review," *Proceedings of the 2021 Americas Conference on Information Systems*, Virtual Conference, August 2021.

17 Hoogsteen, D. and Borgman, H. "Empower the Workforce, Empower the Company? Citizen Development Adoption," *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, January 2022.

18 Viljoen, A., Nguyen, J., Kauschinger, M. and Hein, A. "Fostering Scalable Citizen Development in Organizations: Towards a Guiding Framework," *Proceedings of the 29th American Conference on Information Systems*, Panama City, April 2023.

19 Viljoen, A., Altın, E., Hein, A. and Krcmar, H. "Beyond Citizen Development: Exploring Low-Code Platform Adoption by Professional Software Developers," *Proceedings of the 2024 Americas Conference on Information Systems*, Salt Lake City, August 2024.

20 Scharpf, D., Viljoen, A., Hein, A. and Krcmar. H. "Business Unit Development: Benefits and Challenges for Employees," *HMD Praxis der Wirtschaftsinformatik*, July 2024.

21 Elshan, E., Dickhaut, E. and Ebel, P., op. cit., January 2023.

22 Alamin, M. A. A., Uddin, G., Malakar, S., Afroz, S., Haider, T. and Iqbal, A. "Developer discussion Topics on the Adoption and Barriers of Low Code Software Development Platforms," Empirical Software Engineering (28:4), November 2023.

Table 1: Focus Areas of Extant Literature, Knowledge Gaps and Study Implications (Continuation)

Focus of Extant Studies	Description and Knowledge Gap	Research Need and Opportunity	Implications for Study Focus
Lack of Practical and Proactive Recommendations	The challenges we investigate—software quality, shadow IT and technical debt—have been identified in the literature,[23,24] but these studies have not focused on how to address or mitigate these problems proactively.	Though there are select industry-developed recommendations to govern citizen development,[25] they are high-level and do not provide insights into how citizen development challenges can be addressed or mitigated "practically."	Our study provides concrete recommendations for governance measures to maintain high-quality citizen development.

Study Design and Case Study Partners

To address the shortcomings of the extant literature and to further our understanding of the knowledge gaps, we undertook a qualitative case study involving two firms. The first firm, referred to anonymously as "Alpha," has deployed an external low-code development platform for automating processes and workflows. The second firm, "Beta," is a software provider that has developed a low-code development platform to optimize and manage processes. The study comprised 30 interviews (15 in each firm) with technical experts and citizen developers. (See Appendix A for an overview of our research methodology interviewee profiles.) We were particularly interested in the role that technical experts play in assisting citizen developers in upholding high software standards, as well as the unique characteristics of low-code development platforms (compared to traditional software development) that may inform citizen development governance measures.

Our first round of interviews was with representatives of Alpha. Alpha uses a low-code development platform for workflow automation in/across several business units and functions, primarily for customer service management and IT service management. The interviewees in Alpha included citizen developers, professional developers and other technical experts knowledgeable about the low-code development platform. All citizen developers had business backgrounds and limited technical IT knowledge. We did not limit our interviewees to one specific business unit, which allowed us to obtain insights from various perspectives within the firm.

During the interviews at Alpha, we explored the general challenges of assigning software development tasks to citizen developers and quickly uncovered two insights. First, we found that technical experts play an indispensable role in "jumping in" to guide citizen developers to uphold best practices. Second, we found that the distinct characteristics of low-code platforms—compared to traditional software development tools—determine how citizen development should be governed.

Based on the insights gained from the Alpha interviews, we wanted to home in on the collaboration between technical experts and citizen developers and investigate the characteristics of low-code development platforms, specifically from a provider's perspective. Our second round of interviews was therefore with representatives of Beta, a multinational European firm that provides a low-code development platform to optimize workflows and better manage business processes. The Beta interviewees comprised technical customer support experts who assist citizen developers using the firm's low-code platform. Beta is structured so that customer support representatives have first contact with customers (citizen developers) to address their needs on a day-to-day basis. Depending

23 Elshan, E., Dickhaut, E. and Ebel, P., op. cit., January 2023.
24 Hoogsteen, D. and Borgman, H., op. cit., January 2022.
25 Wong, J., Driver, M. and Saikat, R. *The Future of Apps Must Include Citizen Development*, Gartner, Inc., October 4, 2019, available at https://www.gartner.com/en/documents/3970067?ref=clientFriendlyURL.

on the issues' complexity, customer support representatives can directly assist citizen developers or request assistance from more skilled low-code consultants. These consultants can then assist users with more complex platform-related issues. Thus, compared to Alpha, where we obtained an internal perspective of citizen development challenges, the Beta interviews allowed us to obtain an external perspective on citizen development challenges.

Challenges Arising from Assigning Software Development Tasks to Citizen Developers

This study's primary focus is on providing prescriptive recommendations for effective and efficient citizen development governance to address the three challenges arising from citizen developers' limited technical knowledge. Below, we provide specific examples of the challenges mentioned by our interviewees.

Challenge 1: Software Quality

Both technical experts and citizen developers indicated that software quality may be impacted when citizen developers take on application development tasks. For example, an expert at Alpha noted how citizen developers use overly complex features that impact software performance: "… they would cause [an] impact on the performance because … they were creating … reports with, [special] filters that are too complex, which can cause performance issues." Similarly, another expert at Alpha noted: "[Citizen development has] already impact[ed] [performance]. … We found some slow queries running [on] the platform and [they] are the product of bad[ly] planned reports which they are extracting to use in another application, in another environment." A citizen developer at Alpha also acknowledged that his work might not result in optimal performance: "Just to do the configuration itself is not really a challenge, … but who knows, [there] might be some performance impact. I don't know. I can't tell."

However, even though technical experts acknowledged that citizen developers may produce suboptimal software quality, some also noted that the scope and type of applications that citizen developers produce might not require professional-standard quality. For example, an expert at Beta noted: "I can understand [that] there are risks" but went on to say: "I don't see any bigger concern many times. These kinds of applications are not so risky … I don't feel it's giving too much power to non-prepared people."

Challenge 2: Shadow IT

Both technical experts and citizen developers indicated that without proper oversight, citizen development may pose risks. Specifically, we identified three risks of how citizen development can exacerbate the problems of shadow IT.

First, a key concern—especially emphasized by Alpha—is that citizen developers may make harmful changes in live production environments. A citizen developer at Alpha noted: "… we now have a role which is almost … as powerful as a normal developer role. So, if I am not really very careful … I could damage a few things in the development environment." An expert at Beta noted that, as a low-code platform provider, it tries to foresee problematic implementations by citizen developers: "We would think about an automation [and] if [citizen developers] would [change] something in the source system, or which is not aligned with its owners."

The second risk is that the low-code platform's permission rights might prevent the oversight of applications developed on the platform. An expert at Beta noted that the platform does not always allow experts to review citizen developers' work: "Sometimes, … we can't even get access to the app."

The third risk is that shadow IT problems may arise because of collaboration patterns between technical experts and citizen developers. Citizen developers typically request assistance from low-code platform experts via tickets, providing a structured way for the organization to address and track support requests. However, citizen developers working on similar applications may unwittingly create multiple similar requests for the same applications, leading to experts spending unnecessary time addressing similar requests multiple times.

Challenge 3: Technical Debt

Citizen developers and experts mentioned the implied cost of additional rework caused by the “quick-and-easy” solutions developed. We identified two risks related to technical debt.

The first risk highlighted by interviewees concerns the lack of maintenance of applications. At Alpha, a citizen developer noted: “I don’t maintain the items I have developed. I’m maintaining the items that are in the network team. They could have been developed by anybody. We do not know who developed them, but we are maintaining them, so the items that I create today, … I may not be the one maintaining them.” An expert at Beta similarly noted: “Once they finish developing something, they don’t have people who maintain it.” Moreover, an expert at Alpha noted that citizen developers build applications independently, and there are no measures in place to maintain them: “If the citizen developer … has no replacement, who is supposed to represent him? He basically built it himself, and I think there are a few limits to this.”

The second technical debt risk noted by experts is that widespread, decentralized software development efforts without proper oversight (shadow IT) may lead to the development of duplicate software. An expert at Alpha noted: “Whenever there are too many hands on the platform and most of these hands are not IT people, they tend to recreate stuff … using the [workflow optimizing tool], for example, they create mods for duplicated actions.”

Rethinking and Redesigning Roles Within Software Development

In addition to recognizing the crucial role of experts in aiding citizen developers, a main finding of our study is that a simple binary categorization of citizen developers and technical experts is inadequate for capturing the diverse roles and responsibilities of those involved in citizen development. Thus, before providing concrete managerial recommendations on *what* must be done to mitigate the citizen development challenges, we first explain *who* must undertake such tasks. Additionally, a more detailed distinction between experts and non-IT specialist citizen developers not only enriches understanding of their roles but also illustrates how these roles diverge from those in traditional software development.

Recognizing Two Types of Citizen Developers

We differentiate between two types of citizen developers: “traditional” citizen developers and low-code champions.

Traditional citizen developers: As described earlier, traditional citizen developers are “tech-savvy business users” who undertake application development tasks in their business units where they have domain knowledge and a thorough understanding of the business processes. For example, Alpha’s citizen developers add value by creating dropdown lists or defining condition-based business rules in applications they regularly use.

Low-code champions: This type of citizen developer comprises individuals who are also non-experts but have a deeper knowledge of low-code development platforms. They are the “poster children” or “success stories” of citizen development in their firms because they have successfully taken the initiative and addressed challenges in their business units. Both case study partners indicated that low-code champions could be part of a centralized center of excellence that acts as an innovation hub for disseminating new technologies such as low-code platforms. Moreover, low-code champions can also assist and guide other citizen developers, and thus play an intermediary role between experts and citizen developers. An expert at Beta described low-code champions as those who “act as a bridge between the business and IT” and “that kind of middle person [who] helps you translate things.” This intermediary role has also been extensively documented in established related, domains such as end-user computing and end-user development.[26]

Recognizing Three Technical Expert Roles

We identify three expert roles involved in citizen development processes.: 1) generic low-code development platform support consultants, 2) low-code development platform implementation consultants, and 3) traditional professional developers.

26 Mackay, W. E. “Patterns of Sharing Customizable Software,” *Proceedings of the 1990 ACM conference on Computer-Supported Cooperative Work*, Los Angeles, September 1990, pp. 209-221.

Generic low-code development platform support consultants: These technical experts have specialized knowledge of a platform's capabilities but typically are not responsible for application development themselves. They are the first line of contact for citizen developers and typically assist with simple queries—for example, how to create the logic for a specific attribute, such as selecting specific columns for a table in a report/app. Generic support consultants can either be employed internally in the firm or be located at the platform provider and thus contracted in an external support capacity. In Beta's case, these consultants are external and assist citizen developers daily with simple queries. Such consultants are typically assigned to a broad geographical region and support users in a similar way to typical customer support desks.

Low-code development platform implementation consultants: These are experts on the broader platform architecture who are, however, not responsible for developing applications themselves. Compared to generic support consultants, implementation consultants apply their greater technical proficiency and knowledge to support citizen developers as they develop applications, providing more specific and technical assistance where necessary. At Alpha, implementation consultants are employed internally and go by different names depending on their function and expertise level, such as "platform architects," "success architects," "enterprise architects" and "platform consultants." In Beta's case, implementation consultants are external. Unlike generic support consultants who are assigned to broad geographical regions, implementation consultants at Beta are typically assigned to one or more specific companies and assist citizen developers where queries cannot be addressed by generic support consultants.

Traditional professional developers: These technical experts are knowledgeable about the complex intricacies of conventional software development. At Alpha, only a small number of interviewees could be designated as professional developers. Moreover, we found that this role in the context of citizen development is limited because implementation consultants are responsible for most platform architecture tasks and context-specific implementation topics. At Beta, traditional professional developers are responsible for implementing new features on the low-code platform based on feedback from customer service consultants and technical consultants. However, these developers typically do not interact with the customers (citizen developers) themselves. Table 2 summarizes the roles and responsibilities of the different types of citizen developers and technical experts uncovered in our study.

Citizen Development Roles vs. Traditional Software Development Roles

Based on our classification of citizen developer and technical expert roles, we identify three factors that highlight how role demarcation in citizen development differs from traditional software development.

1. Focus of expertise: The focus of expertise in citizen development is distinguishable from traditional development. Traditionally, experts (i.e., software developers) are responsible for developing software for particular use cases and maintaining the broader software architecture. In citizen development, however, technical experts' expertise shifts much more strongly to context-independent expertise, such as knowledge about the features and architecture of a low-code development platform. Nevertheless, as citizen developers run into challenges, experts such as implementation consultants can still provide implementation-specific assistance, as observed in both Alpha and Beta.

2. Locus of technical expertise: Locus refers to where technical experts are situated and what roles external actors play in a firm's software development efforts. Though software development outsourcing is an established practice,[27] it is typically used for the development of comprehensive software projects for fully fledged applications. However, the ad hoc, customer-service-style inclusion in daily software development workflows that low-code platform vendors offer (e.g., generic support consultants) introduces a new way for firms to approach software development.

27 Chang, Y. B., Gurbaxani, V. and Ravindran, K. "Information Technology Outsourcing," *MIS Quarterly* (41:3), September 2017, pp. 959-973.

Table 2: Overview of Roles and Responsibilities in Collaborative Low-Code Development

	Role	Description	Key Attributes and Responsibilities
Non-Experts	**Traditional Citizen Developers**	Individuals without formal coding experience who build applications using low-code development platforms within their domain of expertise	• Use domain knowledge to identify use cases • Design and develop applications using low-code platforms • Keep up to date with low-code platform features • Test and refine applications within business units
	Low-Code Champions	Citizen developers with deeper knowledge of low-code development platforms (i.e., "expert" citizen developers)	• Promote low-code platform usage within firms • Assist citizen developers with low-code onboarding • Advocate for best citizen development practices • Act as "bridge" between business and IT
Low-Code Experts	**Generic Low-Code Development Platform Support Consultants**	Specialists in a low-code platform who offer assistance not related to specific implementation use cases	• Assist with simple and generic citizen development queries • Serve as the first line of contact for citizen developers • Advise on optimal platform use and application design • If external, not assigned to specific companies
	Low-Code Development Platform Implementation Consultants	Individuals with a higher level of low-code platform specialization who offer implementation-related assistance	• Offer advanced technical support related to low-code platforms • Assist citizen developers with context-specific implementation • Assist with platform architecture and integration issues • If external, assigned to specific companies
	Professional Developers	Traditional software developers with deep coding and system architecture knowledge	• Ensure code and design integrity • If internal, integrate low-code solutions with other systems • If external, build new features into the low-code platform • If external, typically do not interact with citizen developers

3. Tool specificity: In traditional software development, technical expertise may be limited to a particular programming language but is not bound by a specific tool or development environment because, for example, universal programming languages can be used in a variety of different development environments. However, citizen developers and certain low-code development experts' knowledge is limited to a specific low-code platform. Because of the differences between low-code development platforms, citizen developers cannot directly apply their knowledge to a new platform, and low-code consultants may also only have technical expertise on a specific platform. Table 3 summarizes the three factors that distinguish traditional software development roles from those in citizen development.

Table 3: Factors Affecting Role Demarcations in Citizen Development vs. Traditional Software Development

Factor	Traditional Software Development	Citizen Development
Focus of Expertise	Technical experts focus on developing software for specific use cases and maintaining broader software architecture.	Expertise shifts more strongly to context-independent knowledge (e.g., features and architecture of low-code platforms) with a limited focus on specific use cases.
Locus of Technical Expertise	Technical experts are situated within the firm or expertise is provided by outsourcing firms for comprehensive software projects.	Technical expertise can be situated outside the firm even for smaller software projects, with low-code platform vendors offering ad hoc, customer-service-style support.
Tool Specificity	Technical experts may be limited to a particular programming language but are not tied to a specific tool or environment.	Knowledge is limited to a specific low-code platform, and as each platform is different, this necessitates platform-specific expertise.

Recommendations for Governing Citizen Development

The 30 interviews provided rich insights into how citizen development can be governed to reap the benefits of decentralized, rapid software development while mitigating the challenges it brings. Based on these insights, we provide three overarching sets of recommendations for governing citizen development. The first set concerns project, scope and complexity management, the second set relates to organizing effective collaboration, and the third set focuses on education and training. Moreover, we recognize that the recommendations within each set may need to be adapted as citizen development initiatives unfold; thus, we cluster each set's recommendations under the headings of *strategy*, *support* and *control*. Strategy refers to the "big picture" considerations needed to advance long-term citizen development success, support is the organizational and technical resources that managers must provide to empower citizen development stakeholders, and control is concerned with the continuous monitoring of citizen development processes and low-code development platforms, which may, in turn, lead to an adjustment of citizen development strategies or an improvement in support for citizen development. This three-level structure is based on managerial recommendations for one of the predecessors of citizen development, end-user computing.[28]

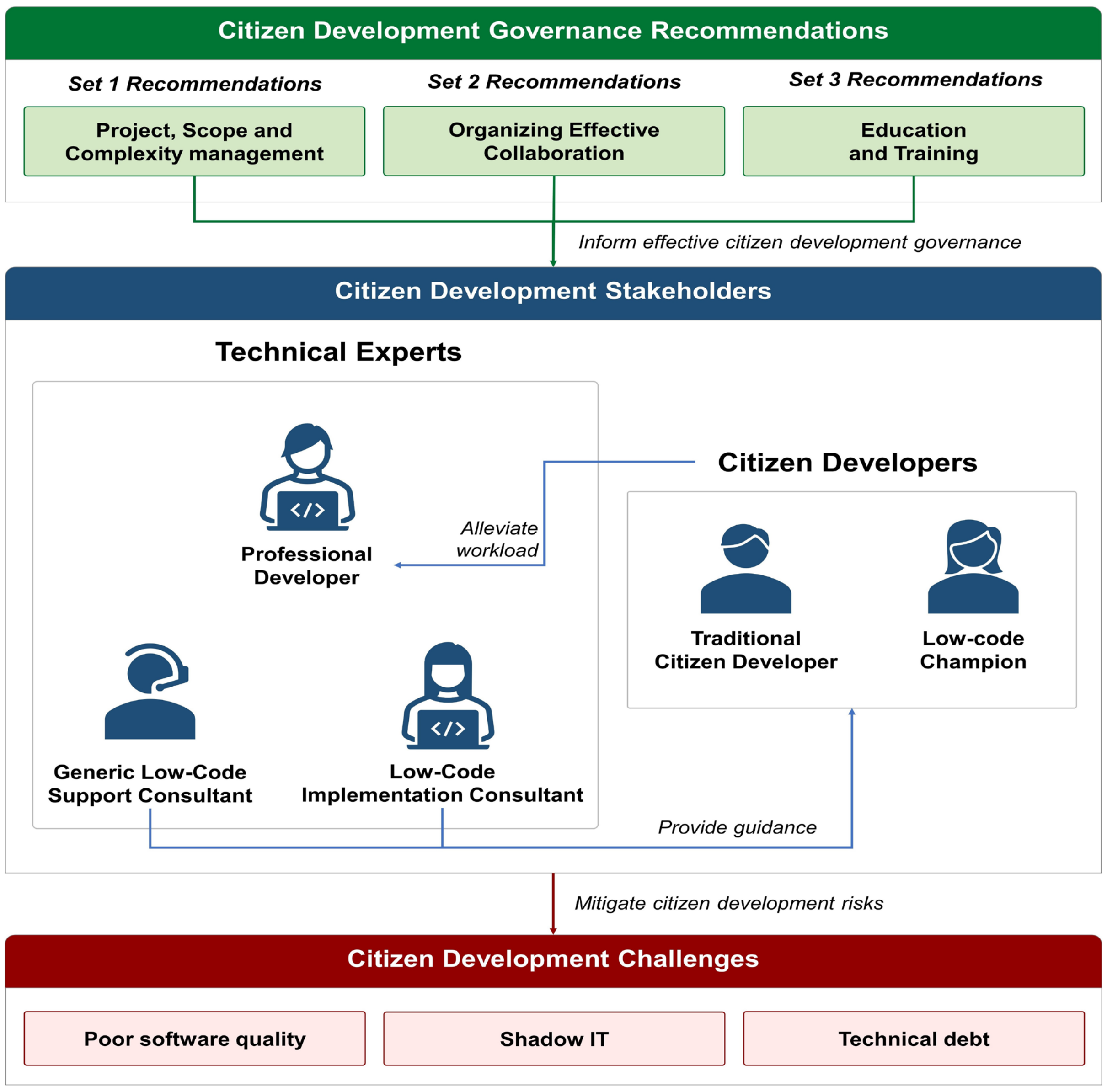
For each set of recommendations, we also provide a table that synthesizes the specific strategy, support and control recommendations, how they mitigate the three citizen development challenges identified above, and which citizen development stakeholders are most affected by the recommendations. A high-level synthesis of the study's results is depicted in Figure 1, showing the relationships between the governance recommendations, different citizen development stakeholders and the citizen development challenges.

Set 1: Recommendations for Project, Scope and Complexity Management

Recommendation 1.1 (strategy)—Select appropriate use cases: Applications developed using low-code development platforms are not complicated and comprehensive software. Rather, citizen developers undertake "simpler" software development tasks that match their limited technical skill sets. For example, a citizen developer at Alpha noted that though he is able to configure simple "routing" within workflows, customizing such functionalities with traditional code falls outside his domain of expertise and is thus more suitable for experts: "What we do here

28 Rockart, J. F. and Flannery, L. S. "The Management of End User Computing," *Communications of the ACM* (26:10), October 1983, pp. 776-784.

Figure 1: Synthesis of Citizen Development Governance Recommendations, Stakeholders and Challenges


Citizen Development Governance Recommendations
Set 1 Recommendations
Set 2 Recommendations
Set 3 Recommendations
Project, Scope and Complexity management
Organizing Effective Collaboration
Education and Training
Inform effective citizen development governance
Citizen Development Stakeholders
Technical Experts
Professional Developer
Alleviate workload
Citizen Developers
Traditional Citizen Developer
Low-code Champion
Generic Low-Code Support Consultant
Low-Code Implementation Consultant
Provide guidance
Mitigate citizen development risks
Citizen Development Challenges
Poor software quality
Shadow IT
Technical debt

is really a pure configuration thing. If it's more technical, [such as] scripts or whatever that's outside of our scope for citizen development; … it's more really in the hands of a technical developer."

Technical experts noted that a practical way to gauge whether citizen development is a suitable approach for a particular project is to define a project's scope and complexity before its commencement through accurate realistic, and detailed requirement specifications. These specifications will not only indicate whether a low-code development platform possesses the necessary functionalities to address the requirements but will also determine if citizen developers are skilled enough for the task.

Recommendation 1.2 (support)—Provide dedicated and appropriate support: The scope and complexity of citizen development projects also affect the degree and type of technical support required to oversee such projects. Because citizen development projects are typically "simple," extensive technical support is not always necessary. Thus, we recommend that organizations institute a mentorship or buddy system within projects, especially by assigning low-code champions. Such a mentor can guide a citizen developer in navigating the development environment, overcoming obstacles and maintaining best practices, albeit in a "novice-friendly" manner.

Nevertheless, the support of technical experts may also be necessary. If so, we recommend limiting the number of low-code platform experts made available to assist citizen developers during a particular project. Though this may seem counterintuitive, we believe it is better to assign limited but dedicated expert support to a project rather than "opening the floodgates" where the line of support and accountability is unclear. One technical expert noted: "We will not put 10 people there who only look after citizen developers … [It] will be a maximum of one or two who will take care of it." This approach ensures that the guidance and technical support are consistent, focused and tailored to the unique needs of each project.

Recommendation 1.3 (control)—Combine traditional project management with low-code platform functionalities: Interviewees highlighted several methods for continuously monitoring citizen development projects, which we have condensed into two complementary approaches. First, technical experts highlighted that traditional software development project management methods like SCRUM can mitigate the challenges that citizen development brings. As explained earlier in the discussion on technical debt, there is often no "backup plan" for applications developed by citizen developers. However, traditional software development project management methods make provision for this, as an implementation consultant at Alpha noted: "Classic developers or OPS teams always make sure that they either have [someone available] 24/7 or at least substitutes, and that's why you have the SCRUM team, where in principle everyone should be able to answer questions somehow."

The second approach for continuously monitoring citizen development projects is to leverage the built-in functionalities offered by low-code platforms—for example, using platform functionalities that integrate with industry-standard project management tracking software such as JIRA.[29] Interviewees especially highlighted low-code development platforms' built-in versioning capabilities. A generic low-code development platform support consultant told us: "I think right now [citizen development] is much better than what it used to be in the past because now [we have] the possibility of versioning." Table 4 synthesizes the specific strategy, support and control recommendations for project scope and complexity management; shows how these recommendations mitigate the three citizen development challenges, and indicates which citizen development stakeholders will be most affected by the recommendations.

Set 2: Recommendations for Organizing Effective Collaboration

The second group of recommendations focuses on the collaboration process between citizen developers and low-code development platform experts and the measures that can be implemented to streamline such collaboration.

Recommendation 2.1 (strategy)—Establish a proper knowledge base: As explained in the discussion of shadow IT, technical experts can spend unnecessary time responding to duplicate or similar support requests from citizen developers. To reduce this workload, we recommend that organizations establish a citizen development knowledge base of the most frequently occurring assistance requests. A generic support consultant at Beta noted: "We try to document [a support request] …, so that if it comes across again, we … [know how to] … solve it." We also recommend applying the 80-20 rule (the Pareto principle): identifying those requests that represent 20% of the effort but address 80% of the support needs. Ideally, documented previous support requests would provide answers to citizen developers' questions; otherwise, the experts can point them to the appropriate documentation. An implementation consultant at Beta noted, that for more specialized issues, "if there is something which

29 *Jira*, Mendix, last modified June 13, 2024, available at https://docs.mendix.com/developerportal/project-management/jira-connector/.

Table 4: Synthesis of Recommendations for Project, Scope and Complexity Management

Recommendation	Stakeholders Involved and Implications for Citizen Development Challenges
1.1 Strategy: Select Appropriate Use Cases	Citizen developers will typically propose the project or use case from their business unit. Depending on the project's scope, a professional developer can review the proposed use case as they are traditionally responsible for developing applications and thus best suited to gauge whether a software development project is suitable for citizen development. By reviewing "reassigned" software development projects, professional developers are aware of the application (thus addressing the shadow IT challenge) and can prevent citizen developers from overreaching their technical competencies, thereby also mitigating the risk that poorly designed or insecure applications are developed (thus addressing the software quality challenge).
1.2 Support: Provide Dedicated and Appropriate Support	A low-code champion should be assigned as the first dedicated support person for a citizen developer and can help to ensure that a citizen developer's application adheres to proper standards (thus addressing the software quality challenge) and is known about in the business unit or firm (the shadow IT challenge). Moreover, generic low-code support consultants and implementation consultants can assist citizen developers with more detailed technical guidance to ensure proper software quality and how to avoid pitfalls that may lead to technical debt. As far as possible, dedicated consultants should be assigned to citizen developers, though this is not always possible (e.g., generic low-code support consultants). The level and type of support and oversight will depend on the scope and complexity of the project.
1.3 Control: Combine Traditional Project Management with Low-Code Platform Functionalities	Both low-code champions and professional developers are well-versed in established software development project management methods and can thus aid citizen developers in documenting their software development efforts (the shadow IT challenge). Moreover, low-code development platforms offer built-in features that can ensure best-practice software development—for example, integrating JIRA functionalities that can help with properly documenting and tracking software development efforts (the shadow IT and technical debt challenges).

is like very, very specific, they [can] reach out to customer support."

The documentation should also be easily understandable by citizen developers, as one generic support consultant at Beta noted that "for citizen developers, it's sometimes hard to find information … because they don't know the exact keywords to [use to get] the information or the things that they need [from the] documentation."

Recommendation 2.2 (support)—Use both low-code development platform and conventional collaboration tools: Interviewees noted that collaboration in citizen development projects can be supported by specific tools provided by the low-code platform, as well as conventional collaboration tools. An implementation consultant at Beta noted: "I would say the collaboration is a very powerful feature [offered by low-code platforms] …, so [you] have the option to… leave notes, leave comments, assign tasks to somebody. You can create your groups. You can create your teams. … [You] can do all that in the [platform] itself."

Experts also noted that industry-standard collaboration tools such as Zoom, Microsoft Teams, Webex, and Slack can be used during citizen development projects. Interviewees emphasized Microsoft Teams because of its inherent focus on establishing teams and collaboration and because it provides access to other documentation relevant to the project: "[Citizen developers] grant us access to their Teams, so we can have a look to better understand what might be the issue." Technical experts also strongly preferred providing support via videoconferencing tools because this removes the need for back-and-forth asynchronous communication such as emails.

Recommendation 2.3 (control)—Implement proper permission rights: In

Table 5: Synthesis of Recommendations for Organizing Effective Collaboration

Recommendation	Stakeholders Involved and Implications for Citizen Development Challenges
2.1: Strategy: Establish a Proper Knowledge Base	Because generic low-code development platform support consultants deal with citizen developers' queries every day, they are best suited to assist in establishing documentation that can address the most common queries. Low-code champions can assist in ensuring the complexity level of the documentation is suitable for citizen developers. Establishing proper documentation is a tried-and-tested way of providing resources to assist developers in maintaining best practices in software development (the software quality challenge). Documenting common issues, guidelines or even the existence of developed applications/functions also mitigates the creation of duplicate support requests or applications (the technical debt challenge).
2.2: Support: Use Both Low-Code Development Platform and Conventional Collaboration Tools	This recommendation applies to all citizen development stakeholders. Technical experts' assistance is necessary to ensure that citizen developers maintain proper software quality. Moreover, collaboration ensures that more internal stakeholders have knowledge of citizen development projects, thus addressing the shadow IT and technical debt challenges. Organizing communication and collaboration effectively among all parties can therefore enhance these benefits.
2.3: Control: Implement Proper Permission Rights	Internal experts, such as professional developers and implementation consultants, can assist in determining the optimal permission rights for low-code platform users. This prevents synchronous development issues on low-code platforms, such as when developers inadvertently overwrite each other's work while concurrently working on the same application. Such errors can cause software quality issues or require unnecessary correction efforts (the technical debt challenge). From the platform provider's perspective, generic low-code development platform support consultants can assist citizen developers with permission-related queries should these not be covered in the knowledge base.

addition to the strategic and support governance measures that can be implemented to organize effective collaboration in citizen development projects, organizations should also implement measures to control and manage the technical difficulties this "decentralized" type of software development may bring. Specifically, we recommend that organizations implement proper permissions and rights to mitigate complications arising from synchronous development on low-code development platforms. An implementation consultant at Beta noted: "Since this is cloud-based, it's very important that you know multiple users are logging in at the same time, and these users can be sitting in different regions right all over the globe." He continued: "It's important that ... an admin [person] ... takes care of all these versionings. They take care of the permissions because otherwise it really becomes a mess." Thus, in line with the recommendations for project scope and complexity management, we recommend that the roles and rights for the project are clearly defined from the initial project scoping. Table 5 synthesizes the specific strategy, support and control recommendations for organizing effective collaboration; shows how they mitigate the three citizen development challenges; and indicates which citizen development stakeholders will be most affected by the recommendations.

Set 3: Recommendations for Education and Training

The importance of educating and training citizen developers is a recurring theme in the citizen development domain.[30] Our interviewees also echoed its importance: the better-equipped citizen developers are with technical knowledge,

30 Bernsteiner, R., Schlögl, S., Ploder, C., Dilger, T. and Brecher, F. "Citizen vs. Professional Developers: Differences and Similarities of Skills and Training Requirements for Low Code Development Platforms,"

the fewer potential challenges that citizen development will introduce.

Recommendation 3.1 (strategy)—Identify both the content and the people: Organizations need to determine the essential knowledge areas that citizen developers must grasp to mitigate poor software quality, shadow IT and technical debt. Our interviewees proposed a multifaceted approach to training citizen developers. First, foundational knowledge of software development is necessary, covering aspects such as the software development lifecycle phases, version control and an introduction to coding standards and best practices. Familiarizing citizen developers with tools like JIRA, Trello, Slack and Microsoft Teams is also important. Second, a firm grasp of relational databases and data modeling is also vital. A technical expert highlighted that "Citizen developers often [struggle] with grasping the interplay between the data layer and user interface components." Third, citizen developers should be equipped with skills in troubleshooting and debugging, so they resolve minor issues themselves. Fourth, citizen developers need a basic understanding of application security, data protection and relevant compliance standards to ensure that the organization upholds high software quality standards.

The organization should identify individuals who are enthusiastic about citizen development, as training is most efficient when focused on individuals eager to learn. These "first movers" play an essential role in citizen development collaboration because they become low-code champions[31] in the organization. As such, they also help alleviate the demand for low-code platform experts because they can assist citizen developer peers without needing to involve experts. From our interviews, we identified four strategies for identifying low-code champions: 1) hosting introductory workshops and providing training on low-code development; 2) organizing hackathons to identify individuals who are motivated to solve problems; 3) identifying continuous learners (i.e., individuals who regularly engage in upskilling, webinars or online courses offered by the organization); and 4) identifying motivated individuals based on discussions with team leads, project managers or department heads.

Recommendation 3.2 (support)—Provide access to formal and informal resources: Whereas the strategic recommendations for education and training deal with the "what" and "who" of educating and training citizen developers, this support recommendation deals with the "how." Organizations should take two actions. First, regarding formal resources, they should provide citizen developers with access to training materials for the content identified in the strategy recommendation above. Certification opportunities in recognized methodologies such as SCRUM can be offered for foundational training on the software development lifecycle and on software best practices from a project management perspective. For platform-specific training, most platform providers offer comprehensive learning materials.[32] However, because citizen developers often feel that the time required to undertake such training is not worth the effort, they must be made aware of the tangible benefits. As an implementation expert at Beta noted, the value becomes clear later: "The trainings work really well. So as soon as [citizen developers] start, … they see the value, and they see that [it was worthwhile]."

Second, organizations can foster informal ways of educating and training citizen developers, such as building citizen developer communities. Unlike formalized training, citizen developer communities emphasize the significance of informal and experiential learning in enhancing the skills of citizen developers. An implementation consultant at Alpha emphasized the importance of such a community: "There will be one or two at most [experts] who take care of it [to] impart knowledge, [but] in the end, we have to hope that there is a good internal community." A generic support consultant at Beta also emphasized citizen developer communities' significance for knowledge transfer: "If the citizen developer has … a dedicated person who knows how to solve [a problem] within the community, they first resort to that resource." To foster such communities, we recommend that organizations: 1) set up structured collaborative environments, such as forums or discussion boards; 2) create

31 Bhangar, S. *Meet Power Platform Champions*, Microsoft, October 4, 2022, available at https://www.microsoft.com/en-us/power-platform/blog/power-apps/powerapps-champions/.

32 *Get Started with Mendix Academy*, Mendix, available at https://academy.mendix.com/link/home

Table 6: Synthesis of Recommendations for Education and Training

Recommendation	Stakeholders Involved and Implications for Citizen Development Challenges
3.1 Strategy: Identify Both the Content and the People	All three technical expert roles can contribute to identifying the necessary technical competencies that citizen developers must learn. Internal professional developers can help identify firm-specific knowledge areas, such as the project management methods employed or the fundamentals of the firm's technology stack. Platform consultants can offer guidance on platform-specific knowledge areas. To identify people who need training, low-code champions are best suited to identify promising citizen developers as they themselves went through citizen development training processes. Having knowledgeable citizen developers can assist in addressing the software quality, shadow IT and technical debt challenges.
3.2: Support: Provide Access to Formal and Informal Resources	Though technical experts can assist in identifying necessary knowledge areas, providing access to resources does not form part of their responsibilities. Management should support project management training, while low-code platform providers should provide access to platform-specific resources (e.g., online learning management systems). Low-code champions and traditional citizen developers can assist in providing informal resources. Internal support can especially assist in mitigating organizational challenges like shadow IT and technical debt, while external low-code platform training can mitigate technical software quality issues caused by the platform.
3.3: Control: Provide Appropriate Access	Internal professional developers and implementation consultants can assist in determining the appropriate access rights for citizen developers. This prevents citizen developers from making unauthorized changes to the live production environment (shadow IT challenge), where they may make changes requiring unnecessary correction efforts (technical debt challenge).

shared repositories containing reusable low-code resources; and 3) promote mentorship programs, pairing seasoned citizen developers with beginners to expedite their learning.

Recommendation 3.3 (control)—Provide appropriate access: Our recommendations for organizing effective collaboration stress the need to implement proper permission rights. However, our interviewees also highlighted that providing appropriate access to citizen developers enhances their learning experience. Though low-code development platforms provide simpler user interfaces than traditional integrated development environments, they still offer various features that citizen developers may not immediately understand. As a professional developer at Beta noted: "Sometimes [citizen developers] are a little bit overwhelmed." Therefore, incrementally introducing citizen developers to advanced low-code platform features can help them "acclimatize" to the platform, easing the learning process while simultaneously mitigating the challenge of making undesired changes in live production environments—i.e., addressing the shadow IT challenge. Table 6 synthesizes the specific strategy, support and control recommendations for education and training, shows how they mitigate the three citizen development challenges, and identifies which citizen development stakeholders will be most affected by the recommendations.

Governing Citizen Development vs. Governing Traditional Software Development

Many of our recommendations for governing citizen development show similarities to governance measures in traditional software development—for example, maintaining proper documentation and using established project management methodologies. However, there are also distinct differences, which fall into four areas.

1. Tool specificity: Unlike traditional software development, where governance measures are mostly independent of specific development tools, citizen development governance measures are strongly affected by the specific functionalities that the low-code platform offers. For example, in traditional software development, code versioning using Github is independent of specific programming languages and development environments. With low-code development, however, code versioning is done on the platform itself. Thus, education and training should include best practices in software development, but with a focus on the functionalities of the specific low-code development platform. Other examples of platform-specific governance are how the platform handles permissions and which project management functionalities it offers.

2. Complexity and technical knowledge: Citizen development governance measures require a lower level of complexity than traditional software development. Though many of our governance recommendations mirror best practices found in traditional software development (such as compiling detailed requirement specifications or maintaining documentation), these recommendations must be implemented to suit the level of citizen developers' technical understanding. For example, our recommendations relating to project, scope and complexity management reflect the fact that citizen developers often misunderstand project requirements. Moreover, governance measures such as maintaining a comprehensive knowledge base and documentation must also consider citizen developers' skill sets. Citizen developers may not be interested in the technical intricacies that interest traditional software developers, or indeed may not be able to understand them.

3. Distribution of governance measures: Citizen development governance is the distribution of governance measures between the organization itself and the low-code platform provider. As explained in the earlier discussion on the differences between the roles of citizen developers and traditional software developers, the ad hoc, customer-service-style support that external experts offer (e.g., generic support consultants) provides a new way for firms to approach software development. Including external experts in daily development workflows requires novel measures to manage this collaboration dynamic. These measures are not needed for traditional software development, where experts are primarily situated within the organization. Another example is maintaining documentation and a knowledge base. Though knowledge base articles concerning company-specific implementations should be maintained internally, general low-code platform documentation (e.g., features and functionalities) should be maintained by the platform provider, similar to how programming language or software documentation is typically maintained externally.

4. Permissions and access control: Citizen development governance measures focus strongly on permissions and access control. For example, the control-level recommendation for education and training emphasizes that citizen developers should be gradually granted access to advanced low-code platform features so as not to overwhelm them with technical complexity. This differs from traditional integrated development environments, where developers immediately have full access to all features. Another example is permissions for versioning and synchronous development. Though permissions for versioning are also relevant in traditional software development (e.g., who may commit code), access control is even more significant in citizen development to prevent technical novices from making unauthorized changes in live production environments. Table 7 summarizes these four differences between traditional software development and citizen development governance.

Concluding Comments

Low-code development platforms are powerful tools that can democratize software development within organizations by enabling citizen developers to develop software solutions rapidly and independently. However, based on the citizen development literature and our case studies, such decentralized and fast-paced development can result in challenges in three specific areas: software quality, shadow IT and technical debt. This study investigates how the distribution of citizen development roles

Table 7: Comparison of Citizen Development and Traditional Software Development Governance

Governance	Traditional Software Development	Citizen Development
Tool Specificity	Governance measures are mostly independent of specific development tools.	Governance measures are strongly affected by the specific functionalities of the low-code platform.
Complexity and Technical Knowledge	Adhering to governance measures requires a higher level of technical understanding.	Governance measures require a lower level of complexity to suit the technical understanding of citizen developers.
Distribution of Governance Measures	Experts and governance measures are primarily focused on structures within the organization.	Governance measures are distributed between the organization and the platform provider, with a greater inclusion of external experts.
Permissions and Access Control	Permissions and access control are important but not as prominent as in citizen development.	Permissions and access control feature strongly, especially to prevent citizen developers from making unauthorized changes.

can mitigate these challenges and provides prescriptive recommendations for ensuring that citizen developers build high-quality applications. In particular, these governance recommendations take into account technical experts' involvement in citizen development and the unique characteristics of low-code development platforms. This study therefore offers insights for practitioners where the extant literature on citizen development studies has fallen short (see Table 1).

Though our study offers many "granular" contributions, there are two overarching key takeaways from this study. First, including citizen developers—i.e., those without significant technical prowess—in software development efforts requires a different approach to governing software development. Citizen development governance measures must be based on the demarcation of citizen developer and technical expert roles and account for the limitations of citizen developers' skill sets. Second, the specific features of the low-code development platform deployed play a significant role in governing citizen development. Low-code platforms offer many features that can assist in managing citizen development projects and maintaining software quality. Moreover, using a single, proprietary platform also has an impact on technical experts' roles. For example, vendor-based experts (such as generic support consultants) may play a more active role in the citizen development process.

A closing thought is whether the extensive governance and "red tape" required for high-quality citizen development may paradoxically undermine the rapid and autonomous development capabilities promised by low-code development platforms. Because citizen development is an emerging phenomenon, it is still unclear what trade-offs firms need to make between speed/decentralization and high software quality. Nonetheless, our study findings show that effective governance of citizen development is essential. Without it, a laissez-faire approach to software development could readily result in pervasive substandard software quality, the proliferation of shadow IT and escalating technical debt.

Appendix: Research Methodology

Following a multiple case study approach,[33] we conducted two rounds of 30 semi-structured interviews with a variety of citizen development stakeholders at two companies. First, we conducted 15 exploratory interviews at a multinational European technology firm (referred to anonymously as "Alpha") that uses a low-code development platform for workflow automation.

33 Yin, R. K. *Case Study Research and Applications*, Sage, 2018.

Profile Overview of Citizen Development Stakeholder Interviewees

Role/Designation in Firm (No. of Interviews)	Firm	Interview Total	Role Categorization
Process/service owner (4)	Alpha	4	Citizen developer
Master platform architect (1), platform architect (1), platform consultant (1), service expert (1), enterprise architect (1), success architect (2)	Alpha	7	Low-code development platform implementation consultant
Platform consultant, senior (5)	Beta	5	
Platform consultant, mid-level (3)	Beta	3	
Customer support consultant, senior (4)	Beta	4	Generic low-code development platform support consultant
Customer support consultant, mid-level (3)	Beta	3	
Professional developer (1)	Alpha	1	Professional developer
Product owner (2), program lead (1)	Alpha	3	Other

The aim of these exploratory interviews was to gain a general understanding of citizen development challenges at the organizational level. From these initial interviews, it became clear that citizen developers require guidance from low-code platform experts. Moreover, we observed that certain platform characteristics affect how citizen development is done within the firm (e.g., that a proprietary, external platform provider is used). To explore these observations in more detail, we conducted a second round of interviews at a low-code development platform provider (referred to anonymously as "Beta") to obtain an external perspective and dive deeper into the collaborative dynamics of citizen development. The table above provides an overview of the 30 interviewees and how we categorized their roles. To analyze the interview data, we conducted multiple rounds of open coding, axial coding and selective coding, and used established qualitative data analysis methodologies.[34]

About the Authors

Altus Viljoen

Altus Viljoen (altus.viljoen@tum.de) is a researcher and doctoral candidate in the Chair of Information Systems and Business Process Management (KrcmarLab) at the Technical University of Munich, Germany, where he also received his master's in management and technology. His research interests include low-code/no-code development platforms, automated machine learning and digital transformation in the automotive sector. His research has been published in journals such as *MIS Quarterly Executive* and *IEEE Transactions on Engineering Management*, and in conference proceedings, including the International Conference on Information Systems, the Hawaii International Conference on System Sciences and the Americas Conference on Information Systems.

Marija Radić

Marija Radić (marija.radic@tum.de) works at Celonis, a global leader in process mining technology and one of the world's fastest-growing software-as-a-service firms. She holds a master's degree in management and technology from the Technical University of Munich, Germany, and a bachelor's degree in information systems and technology from the University of Belgrade, Serbia. Her research interests include low-code/no-code development platforms, process mining, digital transformation and sustainable technology. Marija is passionate about promoting women in science, technology, engineering and mathematics (STEM) and advocates equal education opportunities for all.

34 Gioia, D. A., Corley, K. G. and Hamilton, A. L. "Seeking Qualitative Rigor in Inductive Research," *Organizational Research Methods* (16:1), January 2013, pp. 15-31.

Andreas Hein

Andreas Hein (andreas.hein@unisg.ch) is an assistant professor of IT management affiliated with the Institute of Information Systems and Digital Business in the School of Management of the University of St. Gallen, Switzerland. He received his Ph.D. from the Technical University of Munich, Germany. Prior to joining academia, he was a senior strategy consultant at IBM. His work has been published in journals including *MIS Quarterly*, *Information Systems Journal*, *Journal of Strategic and International Studies*, *European Journal of Information Systems* and *Government Information Quarterly*, and in the proceedings of the International Conference on Information Systems and the European Conference on Information Systems.

John Nguyen

John Nguyen (john.nguyen@tum.de) is an experienced consultant at an international software company and a strong advocate for low-code/no-code development platforms, leveraging his extensive background as a developer and process analyst. He holds a master's in information systems from the Technical University of Munich, Germany. His research interests include digital transformation and scalable platform adoption. John has published papers in various outlets and remains committed to advancing the field of information systems through innovative research and practical applications.

Helmut Krcmar

Helmut Krcmar (helmut.krcmar@tum.de) is Professor Emeritus of Information Systems and Emeritus of Excellence at the Technical University of Munich, Germany. He received a doctoral degree from the Saarland University, Germany, and an honorary doctorate from the University of St. Gallen, Switzerland. His research interests include digital transformation, platform-based ecosystems, information and knowledge management, service management, and information systems for healthcare and government. He is a Senior Scholar and Fellow of the Association for Information Systems. His work has been published in *MIS Quarterly*, *Journal of Management Information Systems*, *Journal of Strategic Information Systems*, *Information Systems Journal* and *MIS Quarterly Executive*, among other outlets.