

June 2024

Strategies for Managing Citizen Developers and No-Code Tools

Olga Biedova

Blake Ives

David Male

Michael Moore

Follow this and additional works at: <https://aisel.aisnet.org/misqe>

Recommended Citation

Biedova, Olga; Ives, Blake; Male, David; and Moore, Michael (2024) "Strategies for Managing Citizen Developers and No-Code Tools," *MIS Quarterly Executive*: Vol. 23: Iss. 2, Article 4.

Available at: <https://aisel.aisnet.org/misqe/vol23/iss2/4>

This material is brought to you by the AIS Journals at AIS Electronic Library (AISeL). It has been accepted for inclusion in MIS Quarterly Executive by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Strategies for Managing Citizen Developers and No-Code Tools

No-code and low-code development tools promise faster development and productivity gains by circumventing the traditional IT bottleneck. These tools pose both opportunities and challenges, especially when used by citizen developers. Based on the no-code experiences of various organizations, we identify the strengths of these tools and the risks, as well as the misalignments between no-code-empowered citizen developers and corporate IT departments. We also provide recommended strategies for managing citizen developers and leveraging no-code tools to enhance organizational agility.^{1,2}

Olga Biedova

College of Charleston (U.S.)

Blake Ives

College of Charleston (U.S.)

David Male

Stellantis & You UK (U.K.)

Michael Moore

Mike Moore & Associates (U.K.)

Addressing the IT Bottleneck with No-Code Solutions

“With Excel you can just go so far. No-code solutions are a superior technical solution and much more scalable. No-code can start off looking like a simple spreadsheet-like solution, but then you can build it in any direction you want—even into an ERP, MRP or CRM application. I have clients who have completely replaced their CRM systems by starting with a no-code application as a prototype and building out from there.” Michael Moore

The above insight from Michael Moore, one of the authors of this article and a consultant with years of no-code experience, highlights the power of these tools in providing agility and responsiveness. For many organizations, no-code tools, together with citizen development are helping to solve the problem of an insufficient number of professional IT developers. (The terms “no-code,” “low-code” and “citizen development” are explained below.) As Moore explains, “There is always a shortage of developers. Development teams keep getting bigger and bigger, the delays longer and longer.”



¹ Gabriele Piccoli is the senior accepting editor for this article.

² The authors are indebted to the senior editor and the reviewers, as well as to the anonymous interviewees who donated both their time and expertise.

The growing demand for digital transformation or augmentation can lead to significant IT bottlenecks and costly delays.³ Though temporary workarounds such as spreadsheets and databases are common, they introduce risks, and hiring professional developers is often prohibitively expensive and may be ineffective because of their lack of business insight and knowledge. As a consequence, organizations are adopting no-code and low-code tools, especially when the business needs are urgent but system requirements are unclear. Uncertainty and unstructured demands are driving the adoption of these tools, which promise faster development and user involvement. Increasingly, the tools are becoming viable options for IT executives prioritizing speed over comprehensive features and security. Yet the decision to adopt such tools is often made by citizen developers outside the IT department, driven by their frustration or the urgent need to deploy IT applications in response to business needs.

In our research for this article, we sought answers to two concerns of IT leaders. First, can no-code development tools provide organizational agility and innovation? Second, for organizations considering no-code tools, with or without the support of citizen developers, how should they proceed?

As described in Appendix A, our findings are based on interviews with 14 individuals in four featured case companies and with eight individuals from other organizations. Because two of the case companies requested anonymity, we refer to all four by pseudonyms: “DealerKyg,” “ICMyContainer,” “LegacyFixt” and “BigFinServ.” (Any overlap or similarity with names of other companies is coincidental and unintentional.) Our research team included a process reengineer with many years of experience with no-code and citizen development and a consultant who guided the no-code initiatives in two of the four case organizations as well as in many other companies.

Before describing the four organizations, each of which has adopted innovative strategies

for deploying no-code tools, we first explain the terms no-code, low-code and citizen development.

No-Code, Low-Code and Citizen Development Explained

No-code and low-code platforms offer fast and affordable tools to revolutionize the development of organizations’ information systems. SAP, a large enterprise-level low-code provider, offers the following definitions:⁴

“Low-code is a method of designing and developing applications using intuitive graphical tools and embedded functionalities that reduce traditional—or pro[fe]ssional]-code—writing requirements. Pro-code writing is still part of the development process, but low-code development offers an augmented and simplified experience to help users start creating quickly.

“No-code is a method that benefits from a similar user experience as low-code, but goes the extra mile by allowing non-technical business users to develop applications without having to write even a single line of code. The main difference between low-code and no-code development platforms lies in how much coding knowledge the user needs. Low-code development platforms require some basic coding skills for users to develop and integrate complex applications, while no-code development platforms require no programming expertise.”

The use of low-code and no-code tools has grown quickly over the last decade, with offerings from many vendors, both large and small. In 2021, Gartner predicted that “by 2025, 70% of new applications developed by organizations will leverage low-code or no-code technologies, a significant increase from less than 25% in 2020”⁵ and “by 2024, 80% of technology products

3 For a good example of using citizen development for digital transformation, see Carroll, N. and Maher, M. “How Shell Fueled Digital Transformation by Establishing DIY Software Development,” *MIS Quarterly Executive* (22:2), June 2023, pp. 99-127.

4 What is Low-Code/No-Code Application Development? SAP, available at <https://www.sap.com/insights/what-is-low-code-no-code.html>.

5 Gartner Says Cloud Will Be the Centerpiece of New Digital Experiences, Gartner, Inc. press release, November 10, 2021, available at <https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences>.

and services will be built by those who are not technology professionals.”⁶ There are dozens⁷ of low-code/no-code tools on the market; the tools mentioned by those we interviewed are listed in Appendix A.

The roots of no-code and low-code development can be traced back to the early 2000s, with the emergence of drag-and-drop tools intended for simplified web development. Low-code/no-code tools are among the latest in a continuum of tools seeking to make computing more accessible to employees who lack expertise in computer programming but have substantial business knowledge.

But applications developed by non-IT employees, often called citizen developers, are not new. Decision support systems and spreadsheets emerged in the 1980s, and employees started to develop their own elaborate macros.^{8,9} Today, the widespread proliferation of spreadsheets created by citizen developers is another compelling justification for easier-to-use, self-documenting, no-code tools.

More recent initiatives, include rapid application development (RAD),¹⁰ robotic process automation (RPA)¹¹ and workflow

automation.¹² The trend toward bring-your-own-device (BYOD)¹³ or bring-your-own-everything (BYOE)¹⁴ into the workplace has also contributed to the democratization of IT. Tools such as PC desktop dashboards¹⁵ provide other means to bring extensive computer powers to non-programming users. More recent tools are cognitive automation¹⁶ and generative AI tools, such as ChatGPT.¹⁷

Our Research Focused on the Use of No-Code Tools by Citizen Developers

In our research, we encountered no-code/low-code tools being used by IT professionals and citizen developers. IT departments were using these tools to compensate for skill shortages,¹⁸ increase productivity of existing staff, lower development costs and deliver systems more quickly. Though many organizations already use low-code tools as integral components of their IT strategy, low-code (as opposed to no-code) product offerings as employed by an IT department lie outside the scope of this article.¹⁹ However, Appendix B provides a brief description of low-code tool usage by IT departments and compares these tools to no-code tools.

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 Crivello, A. *Best No-Code Development Platforms*, G2.com, Inc., available at <https://www.g2.com/categories/no-code-development-platforms#learn-more>.

8 For an early discussion on end-user computing, see Maryam, A., Nelson, R. R. and Weiss, I. R. “Strategies for End-User Computing: An Integrative Framework,” *Journal of Management Information Systems* (4:3), Winter 1987, pp. 28-49.

9 Another rapid application development tool that emerged in the 1980s was computer-aided software engineering (CASE). CASE provided a graphical user interface and assisted users in automating some parts of modeling, coding and documenting. Like today’s low-code applications, CASE was targeted at improving the productivity of traditional developers. For more information, see McKinney, A. L. “A Report on Computer-Aided Software Engineering (CASE),” *Journal of Computing Sciences in Colleges* (6:5), January 1991, pp. 85-92.

10 Beynon-Davies, P., Carne, C., Mackay, H. and Tudhope, D. “Rapid Application Development (RAD): An Empirical Review,” *European Journal of Information Systems* (8:3), September 1999, pp. 211-223.

11 Lacity, M. and Willcocks, L. “Robotic Process Automation at Telefonica O2,” *MIS Quarterly Executive* (15:1), March 2016, pp. 21-35.

12 Stohr, E. A. and Zhao, J. L. “Workflow Automation: Overview and Research Issues,” *Information Systems Frontiers* (3), September 2001, pp. 281-296.

13 Harris, J., Ives, B. and Junglas, I. “IT Consumerization: When Gadgets Turn into Enterprise IT Tools,” *MIS Quarterly Executive* (11:3), September 2012, pp. 99-112.

14 Steelman, Z. R., Lacity, M. and Sabherwal, R. “Charting Your Organization’s Bring-Your-Own-Device Voyage,” *MIS Quarterly Executive* (15:2), June 2016, pp. 85-104.

15 An example of a PC desktop dashboard is Domo (<http://www.domo.com>).

16 Lacity, M. and Willcocks, L. “Becoming Strategic with Intelligent Automation,” *MIS Quarterly Executive* (20:2), June 2021, pp. 169-182.

17 Mondal, S., Das, S. and Vrana, V. G. “How to Bell the Cat? A Theoretical Review of Generative Artificial Intelligence Towards Digital Disruption in all Walks of Life,” *Technologies* (11:2), March 2023.

18 Despite recent layoffs of software engineers by high-tech firms, jobs in software development and testing in the U.S. are forecast to grow by 25% between 2021 and 2031. See *Occupational Outlook Handbook*, Software Developers, Quality Assurance Analysts, and Testers, U.S. Bureau Of Labor Statistics, September 6, 2023, available at <https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm>.

19 For a discussion on the use of low-code tools, see Novales, A. and Mancha, R. “Fueling Digital Transformation with Citizen Developers and Low-Code Development,” *MIS Quarterly Executive* (22:3), September 2023, pp 221-234.

Table 1: Comparison of the Four Case Companies' Experiences of Deploying Low-Code/No-Code Tools

	DealerKyng	ICMyContainer	LegacyFixt	BigFinServ
Level of Low-Code/No-Code Adoption	Very High	High→Low	Very high → Low → ???	Moderate
Company Type	Incumbent (division)	Fast-growing startup	Fast-growing startup	Incumbent (corporate)
Application Areas	Process redesign, (e.g., customer service)	Customer portal	Suite of business applications	AI, rule-based workflow automation, CRM
Low-Code or No-Code?	No-code	Both	No-code	Both
Use of Citizen Developers	Heavy (by full-time citizen developers)	Heavy	Heavy → none (consulting support with no-code tools)	Yes (failed)
Benefits and Opportunities	Speed, cost, business knowledge, talent development	Speed, customer lock-in	Agility, cost, speed, first-mover advantage	Speed
Risks and Challenges	System security, attitude of corporate IT department, IT governance, shadow IT	Vendor sustainability	Vendor credibility with customers, IT governance, architecture	Labor, vendor lock-in
Key Lessons Learned	Hire internal “problem solvers” as citizen developers	Avoid disrupting customers’ work systems	Lack of front-end architecture, agility provided by no-code tools	Staffing, viability of citizen development approach

Our research focuses on the use of no-code tools by citizen developers operating outside their companies' IT departments. These developers, sometimes referred to as “democratized” or “federated” developers, typically possess minimal to no programming experience, aside from basic knowledge of tools like spreadsheets, macros and simple databases (e.g., Microsoft Access). What they often do possess is hands-on experience and a strong understanding of existing work processes. For example, process redesigners at DealerKyng, leveraged no-code development tools to automate, enhance or even reconfigure the company's processes. We focus on no-code tools and platforms because they enable individuals in business units to develop automated workflows. At DealerKyng, these individuals are known as

“solution engineers” or, as we shall later describe, “transformers.”

Overview of the Four Cases

The four case companies vary significantly in nature, as do the applications they have developed using no-code tools and the lessons they have learned. For two of the companies, ICMyContainer and LegacyFixt, no-code tools have proven to be an asset of strategic value; for DealerKyng, the no-code approach has been a key factor in redesigning customer service processes. At BigFinServ, the corporate information systems department has cautiously explored the adoption of both no-code and low-code tools. Despite their proven potential in at least three of these companies, integrating no-code tools with citizen development poses considerable challenges and risks. Table 1 compares the benefits, challenges

and key takeaways that each of these four case companies has experienced in the course of deploying no-code tools (or low-code tools).

As shown in Table 1, we found a wide breadth of application of no-code tools and, particularly at the two startups, observed the positive impact these tools had on business success. The descriptions of each of the four cases below illustrate this diversity while demonstrating the benefits as well as the risks and challenges.

Case 1: Reengineering Customer Service at DealerKyng

“One person runs the entire development of an app. He investigates the problem, picks out the initial requirements, designs the prototype, launches it locally, creates a video for launch and maintains it. He is a developer, tester, maintainer and project manager.” Head of Transformation, DealerKyng

DealerKyng is a single-country car dealer organization within a multinational vehicle manufacturing group. It had experienced significant delays in developing IT applications aimed at automating its business processes to enhance customer service in its dealerships. These process redesigns were stymied by a backlog in the centralized IT department, located in a different country.

Frustrated by the delays, a newly hired process designer used no-code tools to implement, in just a few days, a prototype customer service app for use by employees in a single dealership. Multiple iterations of feedback and revisions followed. The app was successfully implemented throughout the DealerKyng network several weeks later. The prototype evolved into a production app with no support or intervention from the corporate IT group. Six years later, that process designer, now titled the “head of Transformation,” is leading a team of nine “solution engineers” responsible for improvements in both customer service and profitability. No-code tools are at the heart of these citizen developers’ efforts. As one of the solution engineers described: “Drag and drop solutions are amazing. They enable us to build things so quickly, to iterate things so rapidly and to get to a solution so quickly.” There is still no IT department in DealerKyng’s dealership group.

Case 2: Expediting Customer Adoption at ICMYContainer

“Our customers told us they were tired of countless spreadsheets and emails. Our platform gives our customers ONE place to seamlessly collaborate and share data with their partners. We take data out of its previous ‘silos,’ opening the floodgates for improved ways of doing business. Each module and feature of our platform was inspired by a specific customer’s request or pain point.”²⁰

ICMYContainer, a startup, provides tailored shipping container lifecycle data services for businesses in the supply chain and logistics industries. During the COVID-19 pandemic, the price of moving a container of cargo (e.g., from Singapore to Los Angeles) increased by a factor of 10. Delays in shipping also increased, as did uncertainty about when cargo would arrive. Container (and therefore, product) visualization became a major problem for shippers and buyers.

ICMYContainer used no-code tools to develop a product that pulled together milestone location data from container ship companies, ports, customs and satellite companies and sold the resulting container visualization service to shippers, freight forwarders, drayage operators (who transport shipping containers by truck to their final destinations) and freight consolidators. The no-code interface enables the company to customize its container visualization product for each new customer in just a few days, thus requiring little or no modification to a customer’s existing work processes and no need for in-house technical support.

Over time, ICMYContainer realized that there was more and more commonality across customers, so the firm’s skilled programmers developed additional hard code, which has reduced the need for customer tailoring. Thus, though the no-code approach was essential when customer demand was high, it became less so as the firm began to develop analytic products for existing customers and sell those products, as well as data services, to far bigger firms that had their own IT support staff. There were no citizen developers at ICMYContainer.

²⁰ Quote from ICMYContainer marketing material.

Case 3: Supporting Revenue Growth at LegacyFixt

"We started eight years ago with only three people and needed to be agile and fast. We did not know what data we needed to grow the business quickly, but we did know we had to very quickly develop systems that would support the spectrum of our activities. I knew it would take too long doing it the conventional way. Then I came across a no-code solution that gave us the ability to develop applications quickly and to adjust them on the fly. We just did not have the luxury to work on architecture for eight months. We started in 2015 with this system and we are still using it today."
Chief Technology Officer and Co-Founder, LegacyFixt

LegacyFixt, a startup focused on supporting legacy software, has rapidly expanded, doubling in size annually for several years and then continuing to grow quickly. Requiring fast, cost-effective app solutions to support its rapidly growing business, the company adopted a no-code development platform with assistance from a consultant. The resulting no-code apps became the backbone of LegacyFixt's operations.

As the company grew, it brought in new hires to fill new roles, who shifted the maturing company's focus toward long-term viability. The advocate of the no-code strategy, one of the company's founders, willingly handed over IT responsibilities to a newly recruited experienced head of IT. The newcomer, perceiving the limitations and vulnerabilities of the existing IT setup, encouraged division heads to seek out packages from reliable software developers. Though these packages had more features, were international in scope and more bulletproof, they were expensive, poorly integrated and poorly aligned with the company's business model. As a consequence, the new head of IT soon left the company. The founder, who had championed and written much of the no-code solutions, began to have doubts about the effectiveness of transitioning away from no-code solutions.

Case 4: Transitioning Development at BigFinServ

"We started looking at low-code platforms less than two years ago and no-code platforms soon thereafter. We were looking for ways to [gain] new capabilities in a shorter period of time. We liked the notion of reusability and self-documentation within these platforms. I believe that, in certain circumstances, ... less experienced developers [are] capable of delivering high-quality applications and functionality." CIO, BigFinServ

BigFinServ, an incumbent financial services firm, processed large quantities of customers' sales contracts and invoices, thus relieving its customers of huge paperwork burdens. The company's corporate IT department had a staff of over 100. The system development operation was sophisticated and well-managed and, with the arrival of its first CIO five years prior, had been centralized, with the business owners responsible for system implementation success. The firm had also aggressively embraced agile development methods and cloud computing. In the previous two years, it had begun to add both no-code and low-code tools and had close to 20 developers working with those tools.

BigFinServ's CIO told us: "We utilize the same development methodology with the low-code platforms that we do with our custom development platforms." Experiments with citizen developers using the low-code tool had failed, because, in the CIO's opinion, "our less technically savvy business staff may not have the mindset required to logically develop and build applications." Instead, business users engaged with several in-house developers, newly trained on low-code/no-code platforms, and BigFinServ hired several traditional developers from outside, also training them on these platforms, to work alongside several others who already had those skills. Though there had been setbacks, the CIO was confident that these new tools would help BigFinServ more quickly align IT solutions with its changing business needs.

Benefits and Strategic Uses of No-Code Solutions

Our research identified the following advantages of no-code solutions, used either alone or with citizen development.²¹

Circumvent the backlog: Companies use no-code tools to bypass scarce, centrally managed, IT resources, enabling applications to be built swiftly. The chief financial officer of a construction company highlighted the struggle of deploying limited IT resources: “User departments aren’t getting the kind of automation that they want and need to make their jobs easier, faster and more efficient. There’s just not enough IT resources to go around.”

Replace spreadsheets and more: No-code platforms replace cumbersome, manual information systems like spreadsheets and emails. ICMYContainer reported significant reductions in emails and spreadsheets, saving \$100 per container tracked.

Efficient requirements gathering: Used along with agile methods, the no-code approach is effective for prototyping and adjusting to changing requirements. A senior executive of a construction company noted its speed and efficacy in capturing requirements: “A no-code or low-code application gets you 90% of the way there very quickly. You really understand what the users want.”

Cost and time savings: No-code tools reduce development costs and time. A consultant who worked closely with LegacyFixt estimated that there could be a tenfold cost benefit by choosing no-code over packages because the financial benefits of the new system will be captured sooner.

Ease of maintenance: BigFinServ’s CIO saw self-documentation and maintainability as advantages of no-code tools: “No-code is much easier to interpret and to support. I can bring another individual in who didn’t write the application, and [he] comes up to speed [in a] significantly shorter [time] than with custom developed.”

A senior software developer in a software services company felt that the benefits of no-code

came from labor efficiency and responsiveness: “Software developers are expensive; we don’t want them making small modifications in response to user feedback. Instead, we want to get designers, analysts, business people, whoever is the closest to receiving the user’s feedback, to be able to make these modifications.”

Improved alignment and acceptance: The DealerKying case shows that involving work teams in development leads to better alignment with work systems and faster acceptance and adoption of apps.

Employee development: At DealerKying, new career opportunities opened up for workers who transition to citizen developers. BigFinServ’s CIO found that employees, particularly younger workers, were self-skilling: “On their own time, some employees are going out on the web and learning these [low-code/no-code] tools.” At Volvo Group, for example, transitioning the workforce to digital natives is a strategic goal. The head of Digitalization People Services is charged with pursuing this HR development program.²²

Attracting talent: BigFinServ uses cutting-edge no-code and low-code tools, alongside generative AI, to attract talent, emphasizing the importance of investing in advanced technology.

Strategic impact: For startups like ICMYContainer and LegacyFixt, no-code has been crucial for embedding their platforms into customer workflows and supporting rapid growth through adaptable business systems.

Challenges of No-Code and Citizen Development

Though the benefits of no-code development are numerous, the approach poses several challenges. We look first at those that pertain primarily to the use of no-code tools and then those that arise when the tools are placed in the hands of citizen developers.

Challenges Related to the Tools Themselves

Finding talent: Though the no-code approach might be expected to reduce the problem of acquiring staff with IT skills, there is, nevertheless, a learning curve. A senior software engineer from a tech company reflected on his

²¹ For a discussion of the benefits of citizen development in a low-code context, see Novales, A. and Mancha, R., op. cit., September 2023.

²² Ibid.

group's dashed hopes of transitioning some employees to low-code developers:²³ "We set out to broaden the group of people who could use this toolset, but ended up instead relying on more technically qualified people."

Employee resistance within the IT department is also a problem. IT development professionals, already experts in a higher-level language, are likely to view a low-code or no-code assignment as a retrograde step. BigFinServ's CIO described the problems he faced in convincing IT staff to use these new tools and in finding local talent skilled in using them:

"Our existing IT personnel were invited to master these skillsets. Many declined. If they've been in a role for 10 years, they know a language, they know technology, they know our systems. I've been there. I know it's a challenging proposition we are bringing to them. [I tell them] 'If all you want to do is be a COBOL developer for the rest of your life and support a payment system, that's probably not a good long-term plan—things change.' Some did raise their hands, but they were not busting down the gates. [So], we had to find a no-code developer. I searched for someone with that skill nearby, but had to cast a much wider net; we worked with recruiters and six different contract staffing firms. We found someone, but had to agree to let her work remotely."

By contrast, over several years, DealerKyn's head of Transformation successfully recruited, trained and deployed eight "solution engineers," drawn from within the company: "We look for problem solvers. One was a sales advisor with no process background, another has a law degree, another a business degree, another worked in the parts department, again with no process background."

Vendor sustainability and dependency: Low-code tool vendors tend to be large established companies. But providers of no-code tools are often much smaller and may even be

startups. Their small size is a red flag in internal or customer sustainability audits. BigFinServ's CIO described several vulnerabilities he faced with the firm's low-code/no-code vendors:

"It's very different from our home-grown apps that run on our servers. There, when something breaks in the middle of the night, I've got staff on call. These SaaS applications are run in the cloud. When something breaks, I'm completely reliant on them answering the phone. I have service level agreements with my internal clients, and I want our vendors to mirror them in my contracts with them. They don't want to change theirs to fit me, and I don't want to change mine and fit them to my customers. Once you get in bed with them [low-code/no-code vendors], you're reliant on them. I can't get off their platform without rebuilding that application all over again. If, at the next contract renewal, they bump my fees up by 25% or 30%, I'm held hostage."

Standardization: IT executives look for standardized software solutions, such as Microsoft Office or Oracle. These solutions reduce licensing fees and simplify hiring, training and maintenance while enhancing integration and interoperability across system components. Standardization also makes low-code solutions such as Microsoft's Power Platform or Oracle Apex and Oracle Visual Builder attractive options. IT departments will similarly tend to choose a low-code tool with no-code ease-of-use capabilities built-in, as opposed to a stand-alone no-code tool.

Challenges Related to Use of No-Code Tools by Citizen Developers

Once citizen development is added to the mix, the list of risks and challenges grows longer. First, it may be necessary to involve the IT department if citizen-developed apps are to interface with mission-critical or enterprise-wide systems or with the perimeter security infrastructure. Without the IT department's involvement, there can also be problems in scaling, maintenance, integration and governance.²⁴ Such involvement

²³ We have also found—thus far only with informal experiments with two classes of students—that some experienced developers tend to be no quicker in picking up no-code skills, perhaps because it requires an unfamiliar approach to expressing process logic. Thus, they must unlearn as well as learn.

²⁴ SAP, op. cit.

will be made more problematic if the IT department is wary of citizen developers or fears the growth of “shadow IT” organizations that it might one day have to resuscitate. Redundancies in data, systems and skills are other potential challenges, as is ensuring conformance with data protection regulations.

The proliferation of tools outside the IT department: The low cost of no-code tools makes them popular with citizen developers such as DealerKyng’s head of Solution Engineering or the undercapitalized founders of ICMYContainer and LegacyFixt. But, for the head of a corporate IT group such as at DealerKyng’s parent company, products inexpensive enough to be put on a citizen developer’s personal credit card are a proliferation nightmare.

Security and privacy concerns with citizen-developed apps: IT departments will likely have security and privacy concerns about apps built by citizen developers. At DealerKyng, for instance, when the corporate IT department would not provide an API, no-code apps instead emulated human operators to query and update corporate databases.

IT organizations have, often over many decades, developed expertise in both security and privacy requirements. This knowledge must be regularly refreshed and consistently employed. The risks, however, depend on the nature of the application. Apps that link to external networks, update corporate databases, share personal data internationally, are used in highly regulated industries and so on, will require special attention. It is unlikely that citizen developers will have this knowledge and these skills.

The maintenance and sustainability of citizen-developed apps: In addition to concerns about security and privacy, the IT department will also fear that citizen developer innovations will someday become its responsibility. Even worse, the IT department may have to sort out failures that stem from hidden risks associated with the undocumented dependencies of a citizen-developed system—so-called phantom couplings that emerge when an enterprise system

is modified.²⁵ As the IT director of a healthcare organization observed, the consequences of citizen development can be unpredictable:

“Citizen developers have sometimes significantly changed or altered a process that was already in place. Now you have to redocument whatever policies, procedures or processes existed, and alter them to fit with this new one. It also has to align with the organization’s strategy and goals.”

Citizen development architecture concerns: The no-code champion and co-founder of LegacyFixt recognized that he had not given enough attention to data architecture: “We skipped the proper architecture at the beginning. So, we are fixing instead of building from scratch.”

He found that apps quickly developed to address immediate needs often provided a shaky foundation for future enhancements. Apps created at different times by various citizen developers don’t necessarily result in a cohesive system, especially in terms of data design—an area where a database designer’s expertise can be invaluable. Similarly, though no-code tools facilitate the creation of user-friendly interfaces, the insight of a seasoned user-interface designer is crucial for ensuring consistency across interfaces, boosting productivity and enhancing user satisfaction.

Business disruption resulting from citizen developers’ activities: Another concern is the loss of productivity in terms of business disruption when citizen developers are involved in developing and testing apps rather than performing their usual duties. Citizen developers must still be trained to progress up the no-code learning curve and their co-workers need to overcome resistance regarding the need to test and provide feedback on an evolving app in a shaky but live production environment. The founder of one small company we spoke with, self-taught in the use of no-code tools and JavaScript, had spent months harnessing these tools to “make all our spreadsheets disappear.” Though he was a legitimate citizen developer, he

25 Begonha, D., Kopper, A. and Thirakul, T. *Low-Code/No-Code: A Way to Transform Shadow IT into a Next-Gen Technology Asset*, McKinsey Digital, August 19, 2022, available at <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/tech-forward/low-code-no-code-a-way-to-transform-shadow-it-into-a-next-gen-technology-asset>.

Table 2: Application Development Misalignments Between the IT Department and Citizen Developers

Misalignment	Description	IT Department Perspective	Citizen Developer Perspective
Priorities	Allocation of resources	Across many competing projects	Dedicated to one or several problems or opportunities
Timeline	Time for approval and development	Slow and prolonged	Immediate and fast
Prototypes	Purpose of prototypes	Determining requirements	Trojan Horse, production system
IT Oversight	Development controls and safeguards	Defensive, paranoid, conservative	Oppressive, tolerated or nonexistent
Methodology	How apps are developed	Specialization, highly structured or agile, carefully documented	Single developer, little or no documentation
User Involvement	Role of users in app development	Requirements and implementation	Throughout development
Tools	Software necessary for instructing computer	High-level programming language or low-code tool	No-code tool
Scope	Viewpoint of developer organization	Enterprise alignment	Local unit needs
Source of Innovation	How innovations are introduced	Top-down product or system	Bottom-up process and app

wished he had more help: “I’m still learning; what might take me a week or more to do, probably would take a professional developer a day. I wish we could afford one. I’m fortunate to have a first-rate operations manager who frees up some of my time to work on this.”

IT department resistance to citizen development and misalignments: Together, the challenges and concerns described above explain and foster the IT department’s resistance to the “shadow IT” organizations arising from citizen development. That resistance, no matter how justifiable and understandable, presents a major roadblock to capturing the benefits of citizen development. But sometimes that resistance is not justified, as the head of Customer Experience for a utility company complained:

“We developed a [no-code] app. It was working great and we wanted to expand it so other departments could use it too. IT found out about it and told us they were going to hire a developer and bring it in-house. It would have required us to develop

in [no-code] a central customer data repository [that] other departments could use. IT said, ‘You can’t put all our customer data on the Internet.’ So, I told [IT], ‘Look, you are doing the real key stuff. This is just a supplement to what you do and frees you up from this little stuff that we’re doing in a controlled manner. [And] we already have stuff on Google Docs, and God only knows what people have attached to their emails.’ [IT] built it anyway! I’m pretty sure their real fear is that no-code will take their jobs. The head of the IT group told me more than once, ‘If you bring all this no-code stuff in, you won’t need me.’”

The chief financial officer of a construction company provided a more forgiving, albeit colorful, justification for IT department resistance: “The CIO has got so many fires that he’s already fighting. The last thing he wants is a bunch of renegades running around making his life difficult and creating trash can fires around the organization.”

Table 3: Strategies for Governing, Supporting and Leveraging Citizen Developers

Strategy	Description	Actions	Examples
1. Ignore	No oversight by the IT department	None	_____
2. Police	Controlling citizen development	Create citizen development policy and disseminate it; carry out policy audits	Have a policy for deciding when citizen development is unacceptable—e.g., apps with linkages to external servers
3. Incent	Encouraging desired behaviors	Select no-code tools and develop APIs, code modules and templates	Demonstrate no-code tools and provide potential use cases
4. Support	Providing resources to strengthen citizen development	Provide training, use consulting services, build citizen developer community, recognize the role of citizen developers	Create a directory of no-code apps, build a virtual community of citizen developers and hold an annual conference for them
5. Collaborative Leadership	Developing a parallel organization within the IT department	Hire citizen developers, or provide incentives to transition to the role through job rotation and clear career paths	Promote a successful citizen developer to build and lead a parallel team within the IT department
6. Transform	Using citizen development for competitive advantage	Use citizen development to leverage rapid process, product or market innovation	Locate autonomous citizen developers in product manager's office

We found the relationship between IT departments and citizen developers to be laden with potential conflict and, from the perspective of citizen developers, with frustration and anger. Citizen developers were reluctant to describe the IT department as a valued resource or a willing collaborator. The head of Transformation at DealerK yng said that IT was viewed more as a “police force,” albeit a sometimes necessary one: “[IT] looked for holes in the systems and found some security issues.” But, having not developed the systems themselves, the IT department will, understandably, put the security onus squarely in the citizen developer’s hands. He went on to say: “I had to take ownership of the security piece; IT would not own it because they weren’t in control.”

Our research, enriched by the firsthand experiences of two authors in identifying, developing and mentoring citizen developers, reveals that conflicts between citizen developers and IT departments often stem

from misalignment. As detailed in Table 2, we noted differences in priorities, timelines, purpose prototypes, IT department supervision, development approaches, user engagement, tools, organizational reach and innovation sources. Our recommended strategies for managing citizen developers and no-code tools will enable organizations to mitigate these tensions between IT departments and citizen developers.

Recommended Strategies for Managing Citizen Development and No-Code Tools

We offer the following seven recommendations to senior IT leaders—with two caveats from our practitioner authors. One reminds us that “it depends” and describes his approach as “always situational.” The other one said:

"Prescriptions must be tempered by a company's circumstances and resources, as well as the state of its current support for citizen development. In a startup or SME, one person might be the driving force and the only person you need to get on board. In larger companies, it might be the board or policy-setting committee you need to work through. There, the citizen developer might need to be more of a 'freedom fighter,' working around, rather than through, the IT department, and apologizing after the event rather than asking permission."

Both of these authors, with their hard-won experience marshaling and protecting citizen development resources, remain skeptical that there is a viable leadership role for the IT department in managing no-code tools and citizen developers. They remain hopeful, but not optimistic.

1. Assess Readiness for No-Code Tools and Select a Suitable Strategy

Those contemplating no-code tools, with or without citizen developers, must assess the risks as well as the benefits. The selection of tools and developers will be critical in planning either a broad no-code initiative or a specific project. Restricting the use of no-code tools to the IT department may pose minimal risks but will limit the benefits and extend project timelines. Deciding to leverage citizen developers requires a wider assessment, including identifying and addressing the misalignments described in Table 2 rather than eliminating them. This evaluation will guide the selection of a suitable strategy from the six listed in Table 3 based on the organization's current state and goals and determine which actions to take.

2. Select the Right Toolset

The chosen strategy will determine the no-code or low-code tools needed. If you're not pursuing the citizen development approach, you'll likely opt for low-code platforms with no-code enhancements. This option appeals more to professional IT developers, integrates well with legacy systems and meets sustainability criteria during vendor audits. Organizations embracing citizen development should establish a support

office or system to guide citizen developers in selecting approved tools for different applications. Though you may prefer they use low-code/no-code tools from existing vendors, mandating specific tools could negatively affect their motivation, app quality or the successful development and adoption of the apps they build.

3. Identify Citizen Developers

If you intend to use citizen developers, you should identify and recruit those who may have previously been labeled as creating shadow IT risks but have shown initiative by creating IT workarounds. These self-taught individuals will have demonstrated valuable attributes: business familiarity, basic IT solution skills and problem-solving capabilities. They should be recognized, integrated into relevant teams and supported.

When identifying and training citizen developers, you should manage expectations, acknowledging the high potential for challenges and failure. Unlike typical employees who adapt to and excel within set work processes, citizen developers often seek to innovate beyond the constraints of current systems, aiming to improve, overhaul, streamline or even eliminate current processes.

One author, experienced in identifying and training potential citizen developers, outlined the key attributes of successful ones:

"Business sense is the most important attribute in a citizen developer. I see people who are not yet citizen developers but it's often clear to me in the way they approach their work that they are applying 'systems thinking.' They think in terms of data, processes, events as triggers, and capturing key data at appropriate points to support the business process they're working on. They can usually close their eyes and imagine, 'Wouldn't it be great if I could capture that information at this trigger point and use it to support this work?' If you find that kind of person and give them the right tools and then show them what those tools can do ... [they will make] really successful citizen developers. What I don't know, however, is if that is an innate aptitude or something that can be learned."

DealerKyg has recently begun to use a cognitive assessment tool as part of its process for vetting citizen developer candidates. Good interpersonal skills are also essential. DealerKyg's initial vision of transitioning existing employees to the citizen developer role and embedding them in every department has given way to externally hiring full-time developers trained in the use of no-code tools.

The term "citizen" implies that there is a large pool of existing employees who can transition to the citizen developer role. This implication, however, is unrealistic. These are special people with broad skills, innate talents and broad perspectives. A citizen developer at DealerKyg reflected on his position and the opportunities it has given him:

"I value the business understanding I'm getting. On Monday we're mapping out a process with our directors of sales and aftersales; on Tuesday I am testing an app with one of our financial analysts; and on Friday our lead is meeting with one of our dealerships' mechanics and technicians to support their move from paper to a no-code app that we are about to roll out across the dealers. We are fortunate not to be [in an] operational [role]. Our team has the scope to think of new ideas and bring new ideas to the table and then to fruition. But we also get to be the novices in every part of the business, and speak to the experts and try to understand more and more about the business."

In an organization intent on digital transformation, these change agents are on the front line and, we believe, will be among the organization's future leaders.

4. Support Citizen Developers, Don't Over-Police Them

Citizen developers will need training and help with selecting tools, managing security, linking to legacy systems, designing data structures and interfaces, and so on. DealerKyg's head of Transformation, and chief problem solver, described his approach to training citizen developers:

"I got trained and now train others in a four- or five-day boot camp; trainees go away happy as they know what they can achieve. But they have to go back to their day job and their Excel reports, whereas my job is about process improvement all the time."

The citizen developer role can be added to an employee's existing job, but with part-time developers unlikely to be as productive as professionals, app development will likely be slower. However, full-time citizen developers will need support. Recent *MIS Quarterly Executive* articles have reported on two firms, Volvo Group²⁶ and Shell,²⁷ that have created citizen development support organizations. Shell, for example, organizes boot camps and hackathons, offers training, fosters citizen developer communities and recognizes developer achievements.

The operative words in engaging with citizen developers should be providing "help," "guidance" and "support," rather than "policing," "controlling" and "mandating." The IT department can offer substantial support to citizen developers, especially if it positions itself as a facilitator and resource, not as an enforcer. This might involve assigning leadership roles to seasoned citizen developers who have successfully navigated the challenges of collaborating with IT departments. These individuals typically have problem-solving capabilities, systems thinking skills, and the necessary interpersonal and management skills.

However, certain applications, functions, divisions and industries will still require guidelines for, and limitations on, citizen development. Establishing these boundaries, even when they might be somewhat fuzzy, is a crucial early responsibility for the citizen developer support team.

5. Recognize the Role of Transformers

So far in this article, we have described two main citizen development roles: 1) IT professionals, who may need to adjust to or embrace citizen development; and 2) citizen developers themselves, who are motivated by the desire to circumvent IT development bottlenecks so they can automate organizational processes. In

²⁶ Novales, A. and Mancha, R., op. cit., September 2023.

²⁷ Carroll, N. and Maher, M., op. cit., June 2023.

Table 2, we contrast these roles as top-down vs. bottom-up innovation sources. However, there's a third category of citizen developer: process change agents. These individuals often join an organization at a mid-level, possibly fresh from graduate programs or high-tech ventures that didn't pan out. They possess the skills to leverage low-code/no-code tools for "middle-out" process redesign. This category also includes owners of small startup businesses who, due to cost or time constraints, must bypass traditional IT departments.

The appointment of a head of Transformation at DealerKynq indicates that he is an example of a "middle-out" innovation leader—i.e., a transformer. He arrived equipped with skills and previous experience in lean management and process design. He had to quickly augment those skills with knowledge of no-code development, agile development and, in time, computer security. His do-it-yourself enthusiasm was further fueled by major IT bottlenecks and his own manager's frustration with DealerKynq's IT department.

As organizations move aggressively to digitalization, they will increasingly need to harness the skillsets of these transformers. We believe that education for transformers should be at the center of university graduate and undergraduate business and information systems curricula. Knowledge about how a business currently operates is becoming less important than innovative visions of how it can be redesigned—from processes to products to organization design—for survival in a digital world.

6. Protect and Nurture Innovators and Innovations

The journey toward digital businesses, digital products and digital processes poses huge challenges for incumbent organizations and provides startups with exciting opportunities. But, for both, managing innovation is key to their future success. Innovators, or as the head of DealerKynq's Process Reengineering Group calls them, "problem solvers," are invaluable for the bottom-up automation of work processes. They need to be identified, supported and protected.

By contrast, product, channel or other strategic innovations are often driven top-down.

Because digital technology is increasingly at the forefront in driving such innovations, an organization might expect, even require, its IT department to be front and center stage. But IT departments are constrained by legacy systems, tools, skills and project management practices. To break free of these restrictions, some organizations nurture innovators by setting up specialized units or teams that are separate from the IT department for innovation and product management. In other cases, spin-off organizations are set up, often in recognition that digital innovations may cannibalize, and thereby threaten, an incumbent's products. These initiatives will also need to be protected, likely through actions by C-suite executives, not the IT department.

7. Be Prepared for the Challenges of Scaling Up No-Code Applications

The unique demands of digital innovation may initially benefit from the no-code approach and separation from the IT department, but that may change as digital innovations transition to the scaling-up stage. A recent *MIS Quarterly Executive* article²⁸ described how DEVK, a large German insurance company, had created a wholly owned, but otherwise independent, innovation subsidiary. The spin-off's mission was to identify digital products and develop DEVK's first online auto insurance product. As reported in the article, to speed up its work, the innovation team:

"... purchased a comprehensive out-of-the-box framework called 'Insurance Platform.' This framework provided ready-to-use modules for all areas that an insurance company needs. ... all the necessary components were supplied from a single 'one-stop-shop' source and [the team] did not have to spend time selecting and implementing different software components and building connectors to integrate them."

The initiative eventually transitioned from experimental innovation, with a small number of trial customers being serviced manually, to the scaling-up stage, which required reliable

28 Schiffer, S., Mokler, M. and Teubner, A. "Managing IT Challenges When Scaling Digital Innovations," *MIS Quarterly Executive* (22:3), September 2023, pp. 209-219.

functionality and the capacity to efficiently handle thousands of customers. Reliability, profitability and thus cost control were now the driving forces. The scaling-up team, which included only a subset of the digital innovation team, assembled the requisite expertise to create a reliable, high-capacity, bulletproof system. The components required were not unlike those the corporate IT department was very familiar with but which the innovation team had judiciously ignored. Some requirements could now even be met at minimal expense using software solutions covered by DEVK's software licenses.

Nevertheless, the flexible, rudimentary framework—likely with a no-code interface²⁹—that the scaling-up team started with had provided flexibility and the freedom to trial the new online insurance product. As the new product was scaled up, the framework continued to offer value: “The decisive factor in changing from [no-code to coded] modules was that the systems were integrated with the original framework.”³⁰

Scaling up LegacyFixt's products followed a similar path, though software packages, rather than software developed by the IT department were used to replace much of the no-code functionality. However, LegacyFixt's approach of asking all division heads to find software to support their business appears to have fallen short, perhaps due to the lack of a unifying architecture and/or fit with the company's requirements. One of the founders, and the initial no-code developer explained:

“Four years ago, much more of my time and attention was needed on the customer service side. IT was taking too much of my time; I had to build a team. We handed the IT system to the chief financial officer. That's when the conversation started about taking on other systems. We grew the IT staff, and some [of the new] people [said] ‘I used this other system in my previous job; let's use it.’ Those ... decisions ... were opportunistic ... short-term thinking. They are expensive and we have performance problems with

them. It has become an organic mess. But those were the decisions we made and we're going to have to stick with it for the moment.”

But he feared that LegacyFixt's current problems might represent more than just issues with the package suppliers:

“I am worried about the performance. It could be about the providers or it could be our data architecture. We skipped the proper architecture at the beginning. So we are fixing instead of building from scratch. But it might be better to give up agility and move on planning for the future. Nevertheless, I still see no-code in our future.”

One of our authors, the consultant who had begun to support LegacyFixt's no-code efforts in 2018, shared his own assessment.

“The mistake was allowing every department to go with an application of their choosing. There are two or three systems in marketing, a different system in reporting, another in financial. The integration of those systems was an afterthought. Now, we would not do it the same way again. Would I attempt to meet all the systems' business needs with no-code? Probably not. For instance, it would take a lot of time to replicate all the CRM functionalities.”

The founder and initial developer, however, remained enthusiastic about no-code:

“Would I do it again with no-code? Yes, I would... but I wouldn't build them the same way given our understanding of how we need the systems to behave. The platform provided HUGE value by allowing us to scale quickly. The biggest concern I have about the platform ... is [its] ability to scale and deliver the level of performance we need.”

As with DEVK's innovation subsidiary, scaling up at LegacyFixt had revealed the limitations of a no-code initiative. Fortunately for DEVK, the limitations were encountered and overcome

29 The marketing of a similar insurance framework emphasized the merits of using its no-code editor for developing new products and distribution channels.

30 Schiffer, S., Mokler, M. and Teubner, A., op. cit., September 2023.

during the development process. Unfortunately for LegacyFixt, the problems increased as the firm's hypergrowth continued.

Concluding Comments

In each of the four cases reported in this article, the firms used low-code or no-code tools to quickly create digital products and/or automate and redesign business processes. DealerKyng and BigFinServ, both incumbent businesses, represent nearly opposite poles in the way they embraced citizen development and low-code/no-code tools. DealerKyng adopted citizen development because of inadequate IT support from an overburdened, geographically distant corporate IT group. BigFinServ's IT department recognized and was beginning to reap the performance benefits of the tools but took a more cautious (and currently stalled) approach to supporting citizen developers. The two startups, ICMYContainer and LegacyFixt, embraced the no-code approach from the beginning, and it was instrumental in their rapid growth. But as those two companies matured, they began to gravitate away from no-code development, though LegacyFixt is now having second thoughts. The firm's senior management misses both the agility and the custom tailoring that no-code provided but also laments the problems arising from paying insufficient attention to architecture.

Based on our analyses of the four cases and our interviews with representatives from eight other organizations, we provide recommended strategies for managing citizen development and no-code tools. For startups such as ICMYContainer and LegacyFixt, where there are significant customer lock-in opportunities and thus, first-mover advantages, a strategy that fully embraces the use of no-code tools can offer not only competitive advantages but *sustainable* competitive advantages. LegacyFixt and ICMYContainer used no-code tools to quickly get their business and product respectively up and running, allowing each to quickly accommodate an influx of new customers and, over time, to add additional functionality and/or product features. But as both companies discovered, the sustainability that leads to long-term competitive advantage comes not from the tool but from the first-mover advantage the tool enables. And, as LegacyFixt discovered, as the business grows

there is the risk that ineffective architecture design during expedited development could eventually threaten that advantage.

For incumbents, harnessing no-code developments for sustainable competitive advantage may prove more challenging. Though early adopters may gain short-term advantages, no-code technology is readily available, and the initial benefits may prove easy for competitors to replicate. It is only by combining IT capabilities with other distinctive competencies that a sustainable advantage can be achieved.³¹ Incumbents with highly structured processes and systems already in place tend to be resistant to innovation—as are their IT departments, which are highly focused on supporting the status quo.

In incumbents, the wellspring of process design innovations tends to be the business units, where those processes impact workflows. This is potentially a big opportunity for citizen development. A survey by Creatio found that “Up to 30% of processes are not automated at all [and] only 10% of business processes are fully automated”³² Who better to automate these processes than citizen developers who are familiar with the processes and skilled in the use of no-code tools, as was the case at DealerKyng? But citizen developers must be carefully chosen and supported with tools, time and training.

As shown by the LegacyFixt case, no-code tools can quickly and inexpensively provide the flexible development environment often necessary in the early stages of innovation. However, the need to later scale up no-code developments and create a bulletproof solution will require the knowledge and expertise of an established IT department, which may previously have been perceived as a drag on innovation but might now be indispensable.

There are many advantages of using no-code tools in the hands of citizen developers, transformers or innovation units, as exemplified by the DEVK case. But there are also many challenges and risks. However, as illustrated by the DealerKyng and LegacyFixt cases, these problems are not insurmountable. In a rapidly

31 Piccoli, G. and Ives, B. “IT-Dependent Strategic Initiatives and Sustained Competitive Advantage: A Review and Synthesis of the Literature,” *MIS Quarterly* (29:4), January 2005, pp. 747-776.

32 *The State of Low Code/No Code*, Creatio, 2021, p. 17, available at <https://www.creatio.com/page/sites/default/files/2021-05/Report-May.pdf>.

emerging, digital business world that increasingly rewards digital innovation with profits and market share, innovativeness and agility cannot be lightly dismissed. For incumbent firms that are constrained by legacy systems but forced to survive in an ever more digital world, no-code tools can help to overcome inertia.

Appendix A: Research Methodology

Data was collected over 14 months up to August 2023 via interviews with 14 representatives from the four case companies and eight other companies (see the table below, which lists the industry and role of each interviewee). The interviews lasted between 30 minutes and an hour, with some individuals being interviewed multiple times. Approximately a third of the interviews were in person, the rest occurred via Zoom or Teams and were supplemented by a small amount of email data. All but two of the face-to-face interviews had at least two researchers present who subsequently reviewed

their notes. Transcriptions were made of the online interviews. Interviewees had the chance to review their quotes, which were slightly edited for clarity.

The low-code and no-code tools used in the companies we interviewed are listed in the table on the following page.


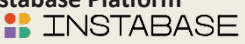






Appendix B: Use of Low-Code Tools by IT Departments

Low-code solutions are often used by IT departments, as they increasingly rely on cloud computing to build apps from globally distributed code modules. Low-code tools are also well-aligned with agile development because they support the main agile principles: enhanced interactions across various organizational roles, focus on the working application rather than extensive documentation, and quick

Summary of the Interviews

Company	Industry	Interviewee Roles	Company Size (Employees)
Case 1: DealerKyng	Car dealership	Head of Transformation Financial analyst Developer Consultant	<5,000
Case 2: ICMContainer	Shipping container logistics	Co-founder/CEO Finance VP Chief technology officer Engineer Professional app developer	<50
Case 3: LegacyFixt	Software support	Chief technology officer and co-founder Consultant	<500
Case 4: BigFinServ	Financial services	CIO	<5,000
Other Interviewees	Healthcare Construction Utility Information systems Consulting Software vendor Healthcare Logistics	IT director Chief financial officer/controller Head of Customer Experience Senior software engineer Low-code/no-code consultant Account executive Cybersecurity director Founder	<5,000 <50 <1,000 >10,000 <10 <50 >10,000 >50

Low-Code/No-Code Tools Used by the Interviewed Companies

Tool Name	Low-Code or No-Code	Application Areas	Cost Tiers
Salesforce Platform 	Low-code	Customer relations (CRM); sales, services, marketing	Platform Starter: \$25/user/month; Platform Plus: \$100/user/month
Instabase Platform 	Low-code	Artificial intelligence; data extraction	Not disclosed
Decisions 	No-code	Rules engine; process and workflow automation	Small business: not disclosed Standard: \$9,900+/month Enterprise: not disclosed
Make 	No-code	General (process automation and application development)	(Depends on no. of operations/month) Core: \$9+/month Pro: \$16+/month Teams: \$29+/month Enterprise: not disclosed
Unqork 	No-code	Financial services, insurance, government, healthcare	Not disclosed
Knack 	Low-code and no-code	General (process automation and application development)	Starter: \$39/month Pro: \$79/month Corporate: \$139/month
ServiceNow's Now Platform 	Low-code and no-code	Workflow automation	Not disclosed
UiPath 	Low-code	Process automation	Pro: \$420+/month Enterprise: not disclosed

responsiveness to changing requirements.³³ According to the Creatio survey discussed above: “Low-code and no-code will amalgamate into a single market segment that will be enterprise-worthy and powerful.”³⁴

Though low-code solutions are being employed outside of IT departments, the special skills required to deploy them are in short supply.³⁵ As a consequence, vendors of low-code tools gravitate toward new interfaces with the ease-of-use characteristics of no-code

tools. With the functionality of the two types of tools increasingly overlapping, low-code tools are being given ease-of-use features like drag and drop and graphical user interfaces, while programming interfaces are often now available for no-code apps. However, vendors of low-code and no-code tools tend to be quite different, as are their price points, target markets and integration with other software. Low-code solutions are targeted primarily at businesses and tend to be integrated with the vendor's other software solutions and marketed to the IT management group. No-code solutions are also targeted at businesses but, as with BYOD, these tools are often acquired by an individual or work group with little upfront expense. As

33 See Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, J. *Manifesto for Agile Software Development*, Agile Alliance, 2001, available at <http://agilemanifesto.org/>.

34 Creatio, op. cit., 2021, p. 19.

35 This is an important trend, as predicted in *ibid*.

a result, an organization might have a small number of judiciously selected low-code tools purchased and used by the IT department and an unknown proliferation of no-code tools scattered throughout the business.

About the Authors

Olga Biedova

Olga Biedova (biedovao@cofc.edu) is an assistant professor of business analytics at the School of Business at the College of Charleston. Her research interests are spread across several areas: emerging information management trends and tools, applied data analytics, portfolio insurance strategies, cybersecurity and scientometric analysis. Olga is a member of INFORMS, the Decision Sciences Institute and the Association for Information Systems. At the College of Charleston, she teaches core business courses (business statistics and management information systems) and business electives (computer-based decision modeling and business analytics).

Blake Ives

Blake Ives (ivesb@cofc.edu) is a Scholar in Residence at the College of Charleston and C. T. Bauer Chair in Business Leadership (emeritus) at the C.T. Bauer College of Business at the University of Houston. He is a senior editor and past editor-in-chief of *MIS Quarterly Executive*, and a past president of the Association of Information Systems. He has also previously been involved with the Society for Information Management, including as the research director of the Advanced Practice Council and as a member and an officer in both the Houston and Dallas chapters.

David Male

David Male (david.male@stellantis.com) is the head of Digital Transformation at Stellantis &You UK, a manufacturer-owned car dealer group. With 25 years of experience implementing technologies in the car dealership sector, he has led many low-code/no-code initiatives to simplify processes and increase efficiency. His expertise in lean thinking has guided his journey from manufacturing to the commercial sector, where he focuses on facilitating change. His practical

application of these solutions has improved operations and agility within the automotive industry and showcases a pragmatic approach to technology-driven change and continuous improvement.

Michael Moore

Mike Moore (mike.moore@complexclear.com) is the managing director of MM&A, a U.K. consultancy that promotes no-code solutions to unlock the potential of citizen developers. With 35 years in IT, Mike has spent the last 10 working with no-code solution platforms like Knack, Zapier and Integromat. His expertise includes integrating tools such as Microsoft Power Apps and Power Automate with business processes for efficiency and innovation. Mike's consulting practice specializes in lean manufacturing methodologies, where he deploys no-code and artificial intelligence solutions to support lean initiatives, including Kaizen and Monozukuri, and in how software technology can facilitate continuous improvement and manufacturing excellence.