MIS Quarterly Executive

Volume 23 | Issue 3 Article 5

September 2024

Combining Low-Code/No-Code with Noncompliant Workarounds to Overcome a Corporate System's Limitations

Robert M. Davison

Louie H. M. Wong

Steven Alter

MIS Quarterly Executive

Combining Low-Code/No-Code with Noncompliant Workarounds to Overcome a Corporate System's Limitations

We explore a novel context where employees engage in low-code no-code work, enabled by software-based workarounds that don't comply with organizational norms and aren't endorsed by corporate management, yet are essential to organizational success. We describe how employees at a warehouse that supports a major retailer in Hong Kong engage in work practices combining low-code/no-code approaches with noncompliant workaround behaviors. We consider the practical implications and potential contributions of such noncompliant work and provide recommendations for business managers overseeing low-code/no-code efforts.[1]

Robert M. Davison
City University of Hong Kong (Hong Kong)

Louie H. M. Wong
Nagoya University of Commerce and Business (Japan)

Steven Alter
University of San Francisco (U.S.)

Non-Platform IT Workarounds Are a Seldom-Reported Mode of Low-Code/No-Code Development

Charles Lamanna,[2] corporate vice president of Microsoft's low-code application platform, emphasizes that low-code development tools can play a transformative role in a variety of organizational tasks, ranging from simple forms and websites to application integration and business process management. A common starting point for organizations keen to adopt low-code approaches is to move away from spreadsheets, not least because they may constitute a security risk. Instead of using spreadsheets, organizations typically migrate data to a professional database, often hosted in the cloud. A cloud-hosted database eliminates the need for multiple account managers to manually update shared drives and simultaneously enables various self-service capabilities. For example, business users can access and update data independently, leveraging the integration capabilities of products such as Microsoft Excel and Access with Office 365 and Microsoft Dynamics.

However, the effectiveness of low-code platforms, such as Microsoft Power Automate, often depends on the underlying application. While Excel remains a powerful calculation tool for

1 Jonny Holmström is the senior accepting editor for this article. We are also grateful to Dr Carroll Noel for his many constructive comments that helped us improve this article immeasurably.

2 Lamanna, C. *New Ways of Development with Copilots and Microsoft Power Platform*, Microsoft blog post, May 21, 2024, available at https://www.microsoft.com/en-us/power-platform/blog/author/charles-lamanna/.

DOI: 10.17705/2msqe.00099

data analysis, Microsoft Power Automate offers low- and no-code solutions for business process automation. Low-code application platforms enable basic user interactions with data, helping to streamline workflows and improve efficiency. Because they leverage digital capabilities to drive automation, these platforms can also be used to automate business processes that previously relied on manual methods, such as email or spreadsheets. Though Excel macros and Visual Basic for Applications[3] have historically facilitated process automation, low-code application platforms now offer enhanced capabilities that reduce reliance on professional developers and meet the evolving needs of businesses in today's digital landscape.

The past decade has seen a surge in groundbreaking big data software aimed at analyzing, manipulating and visualizing data. However, even after 30 years, Microsoft Excel remains the tool of choice for the average knowledge worker trying to make sense of data. It is remarkable that Satya Nadella, CEO of Microsoft, asserts that Excel remains unparalleled among the company's products, with millions of knowledge workers worldwide relying on it every day. He says: "Think about a world without Excel. That's just impossible for me"[4] Excel has served knowledge workers faithfully for decades and plays a pivotal role in business. In fact, this spreadsheet application can be seen as an early example of low-code development. Its user-friendly interface and formidable data analysis, modeling and reporting capabilities have made it an irreplaceable resource.

Historically, non-IT personnel have long played a role in software application development, whether using computer-aided software engineering (CASE) tools or within the wider context of end-user computing.[5] There are many advantages of involving employees in application development, not least their familiarity with the organizational processes that need software support and the shortage of professional application developers. Involving users in application development frees up IT personnel to focus on specialist activities.

Today, low-code/no-code application development involves employees (sometimes referred to as citizen developers) using platforms (variously referred to as low-code development platforms or low-code application platforms) that provide the necessary tools, usually visual, that enable them to develop software applications. Organizations deploy these platforms so that employees can develop applications with the oversight of corporate managers who establish and control the low-code/no-code governance policy. This arrangement allows employees to work on specific tasks under the supervision of responsible managers and have access to regular IT personnel if they encounter problems.

In the era of digital transformation, employee involvement in application development provides business value because it enables employees to contribute directly to that transformation process, thereby mitigating any shortage of professional application developers.[6,7] It is also consistent with the broader dictum of "doing more with less" because, in addition to performing application development work, employees are still carrying out their regular jobs. The organization therefore does not have to employ additional people to develop applications. The low-code/no-code approach thus constitutes a form of "organizational citizenship behavior," where the organization's (non-IT) citizens are active participants.

Prior research has shown that organizational citizenship behavior makes a valuable contribution to organizational productivity.[8] For instance, Nadella and Iansiti suggested that low-code technology "can empower a broad range of 'professional' [software developers] and citizen developers to break down isolated silos and

3 Visual Basic for Applications is an implementation of Microsoft's event-driven programming language Visual Basic 6.0 built into most Microsoft Office desktop applications.

4 Weinberger, M. *Satya Nadella Cannot Imagine the World without This One Microsoft Product*, Business Insider, June 30, 2016, available at https://www.businessinsider.com/satya-nadella-excel-is-microsofts-best-consumer-product-2016-6?r=UK.

5 Cavaye, A. L. M. "User Participation in System Development Revisited," *Information and Management* (28:5), May 1995, pp. 311-323.

6 Carroll, N., Móráin, L. Ó., Garrett, D. and Jamnadass, A. "The Importance of Citizen Development for Digital Transformation," *Cutter IT Journal* (34:3), April 2021, pp. 5-9.

7 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.

8 Davison, R. M., Ou, C. X. J. and Ng, E. "Inadequate Information Systems and Organizational Citizenship Behavior," *Information & Management* (57:6), September 2020, pp. 1-10.

drive innovation across traditional boundaries."[9] Gartner forecasted (perhaps optimistically) that by the end of 2024, 65% of all application development will take place on low-code development platforms.[10]

Though the conventional low-code development platform approach to low-code/no-code work dominates accounts in the literature, this dominance obscures other forms of low-code/no-code work that are not platform-based, and application development work carried out by other types of employees. Examples include those who may not see themselves as citizen developers and those who engage in low-code/no-code work that is not under the control of corporate management. Thus, we contend that there are suitably motivated and qualified employees who choose to engage in low-code/no-code work that provides business benefits but does rely on a low-code development platform. Such work does not adhere to IT governance requirements and is out of sight of (and hence not subject to regulation by) senior management.

This alternative mode of low-code/no-code application development is likely to be more prevalent in organizations than is commonly recognized, particularly in situations where employees create their own workarounds for IT-based solutions, with much of this activity taking place in the shadows.[11] Though this kind of IT work may be viewed by some managers as dangerously subversive[12] its likely prevalence indicates its significance. Specifically, the prominence of this kind of activity reflects the degree to which employees have "agency" (i.e., control and decision-making power over how they work). Instead of having their work constrained by a low-code development platform, they adopt the principles of "bricolage" (i.e., making do with what is at hand) as they create novel solutions to their own problems with whatever tools they are familiar with and can access.

As employees are increasingly becoming technically competent, the opportunities for them to undertake work that used to be the preserve of IT employees also become more frequent. This is not a new phenomenon, but the advent of low-code/no-code platforms has accelerated the trend. Nevertheless, though these platforms can enable significant business benefits, it is misleading to consider them as the only way in which non-IT employees can engage in IT work. For many years, some employees have created their own IT-related workarounds. Such workarounds have many causes and can take many different forms. Technically competent employees are singularly well-positioned to engage in creating workarounds and may neither require nor be able to access a formal low-code development platform to do so.

However, non-platform low-code/no-code work seems to have slipped "beneath the radar" of the literature, and perhaps also that of corporate managers eager to reap the benefits that low-code/no-code offers. Thus, we believe that more attention should be given to non-platform low-code/no-code work, particularly IT-related workarounds. It is not our intention to constrain or dictate how employees work, but rather to alert both managers and employees to the possibilities of this work. To achieve this, this article sets out our findings for the research question we address: *How do employees engage in low-code/no-code work outside formal low-code development platforms to create solutions to the problems they face?*

We explored this phenomenon via a case study of employee workaround behavior in the warehouse operations of a global fast-moving consumer goods firm in Hong Kong, which we refer to anonymously as "CoreRidge." (Our research methodology is described in the Appendix.) Before describing the case, we first provide a primer on low-code/no-code approaches and describe how IT workarounds relate to low-code/no-code development.

9 Nadella, S. and Iansiti, M. *Want a More Equitable Future? Empower Citizen Developers*, Wired, December 9, 2020, available at https://www.wired.com/story/want-a-more-equitable-future-empower-citizen-developers/.

10 Vincent, P., Iijima, K., Driver, M., Wong, J. and Natis, Y. *Gartner Magic Quadrant for Enterprise Low-Code Application Platforms*, Gartner, Inc., August 8, 2019, available at https://www.gartner.com/en/documents/3956079.

11 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.

12 Davison, R. M. and Ou, C. X.J. "Subverting Organisational IS Policy with Feral Systems: A Case in China," *Industrial Management and Data Systems* (118:3), February 2018, pp. 570-588.

A Primer on Low-Code/No-Code Approaches to Application Development

The roots of low-code/no-code development approaches can be traced back to tools developed in the 1980s and 1990s for computer-aided software engineering and rapid application development.[13] Low-code/no-code development can also be seen as an evolution of the customization of commercial application software that end users have long been able to perform, often referred to as end-user computing. In general, low-code/no-code development now refers to situations where non-IT employees are formally empowered to create software applications without the direct involvement of IT professionals, often using low-code development platforms with easy-to-use graphical interfaces.

Developing reliable and high-quality software using the low-code/no-code approach provides significant benefits both to the organization (for instance, in terms of resource use) and to employees. It is much more efficient for employees to develop their own applications immediately rather than wait for the local IT department to have the time and resources to do so. As business professionals, employees have a detailed understanding of their business processes; hence their bottom-up innovation can significantly contribute to the organization.[14] However, many IT staff may doubt the ability of nontechnical employees to develop high-quality, fully functional software applications that also comply with organizational and professional design principles and standards, especially cybersecurity standards.

Most prior research on the low-code/no-code approach has focused on low-code development platforms (also known as low-code application platforms). Such platforms provide an application development environment with all the necessary tools,[15] which are often visual, with point-and-click or drag-and-drop functionality. Through these platforms, employees can rapidly develop and deploy custom applications. Crucially, the platforms reduce or eliminate reliance on formal programming languages.

An increasing number of low-code development platforms are now available. Gartner's 2023 Magic Quadrant for enterprise low-code application platforms[16] assessed 17 market-leading platforms, including Mendix, Microsoft, OutSystems, ServiceNow, Salesforce, and Applan. Many of these provide visual tools so that citizen developers do not need to engage with the more arcane aspects of software coding. A visual interface enables a variety of non-IT employees to engage in application development, which reduces costs associated with digital transformation, and application development more generally.[17]

More broadly, in response to the shortage of digital talent,[18] an increasing number of organizations are embracing the concept of citizen development. This approach empowers non-IT professionals to design, develop and deploy lightweight digital solutions to solve specific work-related problems using the IT tools provided or recommended (or at least tolerated by) core IT units.[19] Because many more people are involved, citizen development has a significant impact on democratizing software development.[20]

Understanding IT Workarounds in the Context of Low-Code/No-Code Development

Workarounds are sequences of actions designed by employees to accomplish task-related goals that do not conform with standard organizational practice.[21] Workarounds may be created by individual employees or groups of employees and may address a temporary

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

14 Elshan, E., Germann, D., Dickhaut, E. and Li, M. "Faster, Cheaper, Better? Analyzing How Low Code Development Platforms Drive Bottom-Up Innovation," *Proceedings of the 31st European Conference on Information Systems (ECIS 2023), Kristiansand, Norway*, 2023, available at https://aisel.aisnet.org/ecis2023_rip/82/

15 Bock, A. C. and Frank, U. "Low-Code Platform," *Business & Information Systems Engineering* (63:3-4), December 2021, pp. 733-740.

16 Matvitskyy, O., Iijima, K., West, M., Davis, K., Jain, A. and Vincent, P. *Magic Quadrant for Enterprise Low-Code Application Platforms*, October 17, 2023, Gartner, Inc., available at https://www.gartner.com/en/documents/4843031?ref=null.

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

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

19 Binzer, B. and Winkler, T. J. "Low-Coders, No-Coders, and Citizen Developers in Demand: Examining Knowledge, Skills, and Abilities Through a Job Market Analysis," *Wirtschaftsinformatik 2023 Proceedings* 17, 2023, available at https://aisel.aisnet.org/wi2023/17.

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

21 Alter, S. "Theory of Workarounds," *Communications of the Association for Information Systems* (34:1), January 2014, pp. 1041-1066.

situation, becoming redundant when the situation changes, or they may persist indefinitely.[22] If they persist, they may then be integrated into regular organizational routines.[23]

The focus of much of the extensive prior research on IT workarounds[24] has been on exploring "shadow IT"—i.e., work that is undertaken by individual employees out of sight (and often without the approval) of their superiors. Such shadow work may be done to avoid overly complex organizational procedures,[25] as a form of resistance or protest against the dictates of corporate management[26] or to overcome the inadequacies of formally provided information systems for the work tasks that employees need to undertake.[27]

However, some researchers have reported on how employees create workarounds in the open—i.e., without any attempt to shield them from or with the tacit approval of their immediate superiors and even corporate management. In these cases, employees may work individually or collectively. For instance, in an early account of workarounds in the print industry, employees adjusted their use of a mandatory print management system to ensure work could be completed within resource constraints.[28]

Workarounds are often associated with corporate systems that are poorly aligned with work processes. In these circumstances, employees may have no choice but to create workarounds that do not comply with corporate standards[29] but provide organizational benefits. Workarounds are often subject to regular enhancement following feedback.[30] The creation of workarounds regularly involves bricolage, where employees "make do with what is at hand" by leveraging a variety of resources in a problem-driven fashion to accomplish their goals. Workarounds that address limitations in corporate systems are likely to become permanent and to be formalized as regular routines.[31] According to Malaurent and Karanasios, workarounds both allow employees to "maintain congruence with their work objectives" and function as "an integral part of the institutionalization" of an enterprise system.[32]

Given the infinite variety of circumstances in which workarounds are created, they can be implemented in a wide variety of ways. However, they commonly involve the use of Microsoft Excel to work around the limitations of corporate software that does not adequately support local needs.[33] Though some pundits view software applications like Microsoft Excel as a form of platform software,[34] we argue that these applications do not really qualify as low-code development platforms because they are neither designed, acquired, nor managed with this purpose in mind.

22 Davison, R. M., Wong, L. H. M., Ou, C. X. J. and Alter, S. "The Coordination of Workarounds: Insights from Responses to Misfits between Local Realities and a Mandated Enterprise System," *Information & Management* (58:8), December 2021, pp. 1-12.

23 Pentland, B. T., and Feldman, M. S. "Designing Routines: On the Folly of Designing Artifacts, While Hoping for Patterns of Action," *Information and Organization* (18:4), October 2008, pp. 235-250.

24 Ejnefjäll, T. and Ågerfalk, P. J. Conceptualizing Workarounds: Meanings and Manifestations in Information Systems Research, *Communications of the Association for Information Systems* (45:20), 2019, pp. 340-363.

25 Beerepoort, I., van de Weerd, I. and Reijers, H. A. "The Potential of Workarounds for Improving Processes," *Proceedings of the International Conference on Business Process Improvement*, September 1-6, 2019, Vienna, pp. 338-350.

26 Davison, R. M. and Ou, C. X. J. "Digital Work in a Digitally Challenged Organization," *Information & Management*. (54:1), January 2017, pp. 129-137.

27 Davison, R. M., and Ou, C. X. J., op. cit., February 2018.

28 Bowers, J., Button, G. and Sharrock, W. "Workflow from Within and Without: Technology and Cooperative Work on the Print Industry Shop Floor," *Proceedings of the 4th European Conference on Cooperative Work*, 10-14 September, 1995, Stockholm, Sweden, pp. 51-66.

29 Alter, S. "Beneficial Noncompliance and Detrimental Compliance: Expected Paths to Unintended Consequences," *Proceedings of the 21st Americas Conference on Information Systems*, Fajardo, Puerto Rico, August 13-15, 2015.

30 Safadi, H. and Faraj, S. "The Role of Workarounds During an Open Source Electronic Medical Record System Implementation," *Proceedings of the 31st International Conference on Information Systems*, December 12-15, 2010.

31 Malaurent, J. and Avison, D. "Reconciling Global and Local Needs: A Canonical Action Research Project to Deal with Workarounds," *Information Systems Journal* (26:3), May 2015, pp. 227-257.

32 Malaurent, J. and Karanasios, S. "Learning from Workaround Practices: The Challenge of Enterprise System Implementations in Multinational Corporations," *Information Systems Journal* (30:3), November 2019, pp. 639-663.

33 Spierings, A., Kerr, D. and Houghton, L. "Issues That Support the Creation of ICT Workarounds: Towards a Theoretical Understanding of Feral Information Systems," *Information Systems Journal* (27:6), September 2016, pp. 775-794.

34 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.

The CoreRidge Low-Code/No-Code Case

Background to CoreRidge and Our Research

With operations in over 50 countries, CoreRidge focuses on household retail goods and is a global player in the fast-moving consumer goods industry. Prior to 2010, each of CoreRidge's 400+ locations was free to operate an enterprise system of its own choice or design, but in the following decade, the company introduced a standardized ERP solution that relied heavily on a product now known as Microsoft Dynamics 365 Business Central. CoreRidge opted to implement a "plain vanilla" version of the software that did not allow any customization or modification. As a result, major discrepancies emerged in some of CoreRidge's operating locations between employees' prior work conventions (i.e., approved business processes) and the way that work could be done following the implementation of the new software procedures.

We explored this situation in one of CoreRidge's five operating locations in Hong Kong (a warehouse and four retail stores), where it is the market leader in terms of sales volume of household goods. We focused our attention on the warehouse that services the retail stores. The warehouse is responsible for arranging for the procurement of new stock and the distribution of stock to both the retail stores and to individual customers who have requested home delivery. In the course of our analysis, we uncovered that many employees were actively engaged in low-code/no-code application development, as described below.

Limitations of Microsoft Dynamics for CoreRidge's Hong Kong Operations

CoreRidge implemented the Microsoft Dynamics software globally between 2010 and 2016 and found the software was a good fit for many of the company's operating locations. However, due to some unique features of the Hong Kong environment, it did not fit many of the local business processes in Hong Kong, which were designed to cater to local customer requirements and constraints and, as such, cannot easily be modified. For example, in most countries, CoreRidge's retail sites are physically co-located with a warehouse, so that customers can take their purchases home directly from the warehouse or contract with a local delivery operator. In Hong Kong, however, a single warehouse served four geographically dispersed retail sites (during our study period). The reason for this different arrangement is due to the high cost of land in Hong Kong. Retail sites occupy cramped, downtown locations where there is no physical space available for a warehouse. All large items are therefore kept at a central warehouse and dispatched from the warehouse to customers. This is a nonstandard mode of operation for CoreRidge, but it is unavoidable in Hong Kong.

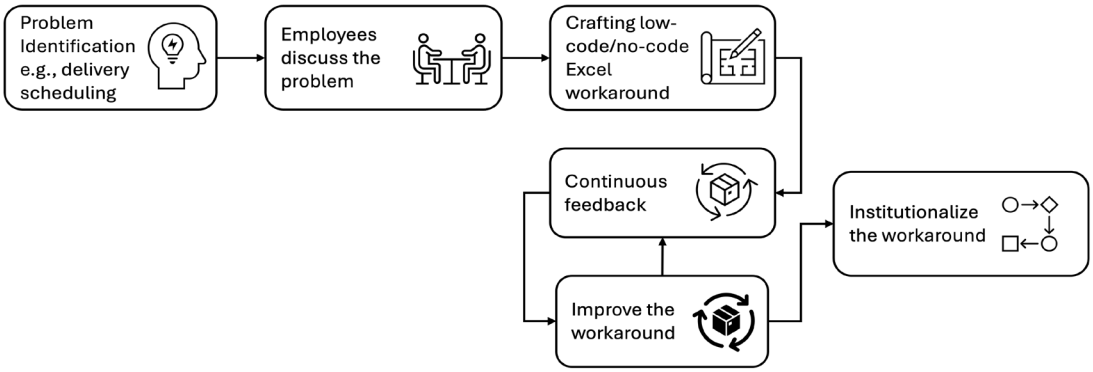
Though Microsoft Dynamics does include a delivery module, it was not enabled by CoreRidge because it could not be modified to handle CoreRidge's situation in Hong Kong. As a result, all deliveries are made through third-party contractors and have to be managed outside the formal system. Further complications arise when customers request a change in the delivery date or when inclement weather (e.g., typhoons or rainstorms) forces all deliveries to be postponed. These situations cannot be handled by Microsoft Dynamics, so employees in Hong Kong have to create other ways to get their work done. Because CoreRidge does not provide a low-code development platform for employees to use, they have to explore other options to ensure that goods can still be delivered to customers.

Creating Excel-Based Workarounds

To solve these local problems, employees create and maintain workarounds with Microsoft Excel that ensure that work can be completed as scheduled (see Figure 1). These workarounds apply mainly to functions that are central to warehouse operations, mostly relating to delivery scheduling and rescheduling. The workarounds that employees create are important because they relate to customer satisfaction: Customers expect to receive the items they pay for.

The Excel-based workarounds are consistent with low-code/no-code principles: They are created by employees and focus on specific tasks that require additional information systems support. Moreover, the workarounds are implemented through simple formulas and functions or drag-and-drop actions such as in

Figure 1: Process Model for Creating, Enhancing and Institutionalizing Workarounds at CoreRidge



Excel's PivotTable, and do not involve any Visual Basic for Applications, macros or other advanced Excel programming techniques.

However, the Excel workarounds do not comply with corporate policy: The global CIO expects that Microsoft Dynamics will be used for all activities. Many employees told us they understood that their use of Excel does not comply with the corporate IT policy but said they had to create Excel workarounds because they could not complete their work with Microsoft Dynamics. As one business navigator noted: "We insist on using workarounds because we just want to get the job done."

Commenting on how workarounds are developed, a goods flow manager explained "When there is a problem, we will have a meeting to discuss how to tackle it. We decide on the workaround with our team. If the impact [of the workaround] is good, we will keep going; if not, we will see how to improve it." Similarly, a business manager noted: "I need to do manual monitoring work on orders to ensure there are no missing or wrong orders. We communicate in the same team, ask other colleagues for their opinions and create some workarounds to make improvements. We use workarounds because the [Microsoft Dynamics] system can't fit our operation."

The workarounds described to us are first discussed by all relevant team members and then implemented with Excel. Several of the employees have extensive experience using Excel, including knowledge they acquired in their university studies. The workarounds involve downloading data from Microsoft Dynamics to Excel and manipulating it in Excel to meet local needs. For instance, Excel is used to create delivery lists of furniture items to customers. It is also used if customers want to reschedule deliveries. Excel is the preferred tool for creating workarounds because it is readily available (CoreRidge has a site license for the software) and easy for the employees to use.

The need to create workarounds to undertake work that Microsoft Dynamics cannot perform is widespread across different functional areas within CoreRidge's warehouse. Workarounds are created for inbound and outbound logistics, forecasting, planning, quality control inventory management and other activities. A customer delivery supervisor reported how he "draws some raw data from Microsoft Dynamics and does forecasting in Excel." He also noted that 100% of his work requires workarounds of some form, the vast majority using Excel.

The nature of the Excel-based workarounds was illustrated by a product quality specialist, who reported: "After information gathering and sorting in Excel, we can accomplish the arrangement for delivery, the forecasting and the number of trucks needed. … The design formulas are based on the contract, like the requirement of delivery service, the number of goods to be delivered, the places to go, the money to be paid to the drivers. The formulae are based on my understanding of the work that needs to be done so that we can input the correct data into Excel.

… Without Excel, there is no way to carry out the delivery of goods."

In a similar vein, the customer delivery coordinator reported that "most routine jobs are done with workarounds using Excel. We use Excel to bring together all the data that we extract from Microsoft Dynamics, and then analyze it."

The central role of Excel in CoreRidge's delivery process was further illustrated by the fact that the employees have developed their own standard operating procedures and even a training manual for Excel-based workarounds. These procedures ensure there are appropriate handover mechanisms when new employees are brought on board and that relevant Excel knowledge is not lost when employees leave CoreRidge.

Using Excel workarounds allows employees to have considerable freedom in how they operate, far beyond what they might achieve if they were using the official low-code development platform. Collectively, employees determine how to craft workarounds, use a low-code/no-code type of approach for the associated technical development in Excel and provide training to their fellow employees. Finally, they document the workarounds and the Excel techniques involved in creating them in an operating manual, thus enshrining their future agency to continue this essential work.

Analysis of the CoreRidge Case

None of our interviewees mentioned the term low-code/no-code development spontaneously, yet their practice of crafting workarounds with Microsoft Excel to solve work-related problems clearly corresponds to the contemporary understanding of the approach. However, unlike many prior investigations of the low-code/no-code approach, these employees do not use a low-code development platform to support their efforts, nor indeed is their work endorsed or facilitated by CoreRidge's management. But the lack of such a platform is by no means a constraint. Instead, employees have the agency and freedom to create Excel-based workaround solutions that neatly solve their problems with minimal fuss or overhead. They work collaboratively on situation assessments, decide how to employ Excel and then put their plans into action. We see the process of employees creating Excel-based workarounds following a low-code/no-code approach as a natural reaction to a corporate ERP system that inadequately fits their work requirements.

CoreRidge's employees do not seem to be concerned about the fact that their Excel-based workaround solutions do not comply with the corporate IT policy because the workarounds are the only way they can complete their work effectively. Indeed, with the passing of time, the workarounds have become institutionalized as standard operating procedures in their own right. However, we recognize that these workarounds may potentially be harmful to CoreRidge because, even though they are approved by the warehouse manager, they are created without any scrutiny by IT personnel and without the use of a corporate-endorsed low-code development platform.

We are not suggesting that the employees would deliberately introduce errors or security flaws, but the risk that they might do so inadvertently is likely to be of concern to the CIO and chief security officer, who are always on the lookout for opportunities to enhance corporate cybersecurity and to penalize noncompliant behavior. For instance, if employees introduce data errors when downloading data from Microsoft Dynamics to Excel, these errors could result in incorrect deliveries to customers. Likewise, customer data (e.g., name and address) could be entered incorrectly. Because there is no Microsoft Dynamics-to-Excel interface, the downloading and editing procedure is necessarily messy and the possibility of errors cannot be ruled out.

In Hong Kong, the use of Excel-based workarounds is restricted to CoreRidge's warehouse, and many employees confirmed that they create or use the workarounds. However, we were informally told about employee-driven workarounds in other CoreRidge operating locations. There are similar practices in Taipei and New York, which are characterized by similar geographical constraints.

Indeed, the literature on workarounds suggests that the practice is widespread. As a result, even though we were unable to find any other published studies documenting situations where employees created low-code/no-code

workarounds without the use of a low-code development platform, we must assume that similar practices are common. We believe that there are many other examples of employees using a bricolage approach to leverage whatever software tools they can access to solve problems. In particular, the use of Excel to replace enterprise system functions is almost ubiquitous. Excel is very much the tool of choice, with its readily available online help system, proliferation of online resources to help users and its long history of use in organizations.

The literature on low-code/no-code development almost universally asserts the positives of the practice. Indeed, the call for papers for this special issue of *MIS Quarterly Executive*[35] highlights the value that citizen developers can bring to their organizations. However, the CoreRidge case shows that platform-based low-code/no-code development is not the only form of the practice; non-platform low-code/no-code development practices also exist and create value but operate in a less transparent space where the integrity of employees is critical. Such unregulated low-code/no-code development—i.e., taking place outside the oversight of management—has the potential to cause significant harm to the organization.

Undoubtedly, low-code development platforms have many strengths, but they may not be appropriate for organizations that do not recognize the need (or even the possibility) to democratize systems development (or do not want to spend money on it). Typically, the objective of deploying a low-code development platform is to provide limited flexibility within a tightly controlled environment, using the platform to enhance organizational mission-critical applications. The scope of these applications is restricted to the work system as determined by the company's headquarters.

Our interviews uncovered no evidence that CoreRidge has any interest in democratizing software development, even though it could be argued that such democratizing is taking place. Indeed, it is doubtful whether CoreRidge's employees really want to be citizen developers—most of them only care about why the organizational mission-critical applications do not support the local processes in their day-to-day operations and what they can do about this. The need for low-code/no-code development is recognized neither by CoreRidge's global CIO nor by company headquarters.

The CoreRidge case shows that in contrast to deploying a corporately mandated low-code development platform, Excel is both more practical and more cost-effective. Excel is ubiquitous and does not require extra investment. Most employees are already familiar with Excel and the required training can be undertaken by the more competent employees. In this way, employees can collectively retain and enhance their own agency, retaining control over all aspects of the low-code/no-code process as they create workarounds.

The creation and use of Excel-based workarounds at CoreRidge involve many low-code/no-code practices but without using a vendor's low-code development platform. The Excel applications created by employees are supported by the corporate ERP software and are used to make business operations more efficient and effective. Tables 1 and 2 explore the concept of platform-less development of workarounds in more depth by using the nine elements of the work system framework[36] to compare typical low-code/no-code assumptions with actual practices in the CoreRidge case. Table 1 compares application development practices in CoreRidge with typical low-code/no-code assumptions related to application development. Table 2 compares CoreRidge's operational work system with typical assumptions about work systems that might use low-code/no-code-based software. Comparing Tables 1 and 2 reveals the difference between using low-code/no-code to perform programming tasks and using the resulting software in operational systems.

A comparison of Tables 1 and 2 highlights an important issue identified in our analysis of the CoreRidge case. Merely using a low-code development platform does not guarantee that operational tasks will be done well. There is no reason to believe that such a platform would provide CoreRidge employees with the opportunity to develop superior workarounds to meet their needs and satisfy both customer needs

35 Carroll, N., Holmström, J. and Matook, S. "Call for Papers: The Rise of Low-Code/No-Code: Accelerating Digital Transformation," *MIS Quarterly Executive*, June 2023.

36 Alter, S. "Work System Theory: Overview of Core Concepts, Extensions, and Challenges for the Future," *Journal of the Association for Information Systems* (14:2), February 2013, pp. 72-121.

Table 1: Typical Assumptions Related to Using Low-Code/No-Code Vs. Actual Development Practices in the CoreRidge Case

Work System Element	Typical Assumption Related to Low-Code/No-Code Development	Development Practices in the CoreRidge Case
Technology	A low-code development platform is used.	Excel suffices. No low-code development platform is available or needed.
Information	Relevant information will be defined using low-code/no-code capabilities, including a small-scale data management system.	Relevant information is defined in column headings of Excel spreadsheets.
Participants	Low-code/no-code capabilities are used by business professionals or IT professionals. Business professionals do not need extensive training before using the capabilities to tailor software to their needs.	Business professionals collaborate in using Excel to produce and maintain spreadsheet-based software that fits their business needs.
Activities	Analysis prior to using low-code/no-code capabilities is assumed to be simple. Use of the capabilities makes the programming effort simple. Business professionals are able to use the capabilities for simple programming tasks and also are able to use them to test their programming.	Most of the spreadsheet development was done collaboratively and incrementally because the Excel workarounds touched so many parts of the warehouse operations. Problems revealed in the use of Excel-based workarounds are discussed and fixed collaboratively.
Product/ Services	A software system is built using low-code/no-code capabilities and tailored to the situation.	Excel-based applications are used by warehouse employees. Creation of standard operating procedures and a training manual for Excel-based workarounds.
Customer	Work system participants benefit from using the software, as does anyone else in the organization who uses the software.	Warehouse employees can do their work efficiently and effectively by using the Excel workarounds.
Environment	Corporate approval for using low-code/no-code capabilities to create software applications.	Corporate (global) expectation that vanilla ERP capabilities would be used. Local recognition that the ERP capabilities did not fit. Local collaboration in developing and using Excel-based workarounds.
Infrastructure	Low-code/no-code tools are part of the corporate infrastructure.	Microsoft Office and the ERP system are part of the corporate infrastructure. Insufficient number of IT professionals for IT-infrastructure roles.
Strategy	Use low-code/no-code to program simple systems without professional IT help.	Use Excel to develop workarounds without professional IT help.

and corporate information needs. On the contrary, the limitations of specific low-code development platforms might make their use by CoreRidge impossible or futile. These limitations might even create difficulties for CoreRidge—for example, in their treatment of processes, data definitions and user interfaces.

Recommendations for Managing Low-Code/No-Code Efforts

Based on our analysis of the CoreRidge case, we provide five recommendations for business

Table 2: Typical Assumptions Related to Operational Work Systems That Use Low-Code/No-Code-Based Software vs. Operational Work Systems in the CoreRidge Case

Work system Element	Typical Assumptions Related to Operational Work Systems That Use Low-Code/No-Code-Based Software	Operational Work Systems in the CoreRidge Case
Technology	A low-code development platform is used.	Excel workarounds are used for some tasks and corporate ERP is used for other tasks.
Information	Information defined in the low-code development platform software is stored in a small-scale data management system.	Relevant information is stored and updated on spreadsheets that are not attached to a data management system. Other information transmitted to corporate headquarters is stored and updated in the corporate ERP system.
Participants	Participants in the target work system use low-code/no-code-based software.	Warehouse employees use Excel-based spreadsheets while performing some of their tasks involving logistics, forecasting, planning, quality control, inventory management and other functions. Some employees use ERP capabilities to communicate information to and from headquarters.
Activities	Participants in the target work system perform tasks using low-code/no-code-based software.	Warehouse employees use Excel-based spreadsheets while performing some of their tasks involving logistics, forecasting, planning, quality control, inventory management, and so on. Some employees use ERP capabilities to communicate information to and from headquarters. Data maintained on spreadsheets is checked extensively.
Product/ Services	Completion of tasks that use low-code/no-code-based software.	Excel workarounds previously produced all information needed to operate the warehouse, coordinate with local shippers, coordinate with headquarters for inventory replenishment and provide financial information required by headquarters.
Customer	Internal and external customers use the information and other outputs produced by the target work system.	Internal and external customers use the information and other outputs produced by the warehouse.
Environment	Relatively stable systems that encounter few exceptions or interruptions.	Extensive cooperation to continue meeting customer and corporate needs despite inadequacies of the corporate ERP system.
Infrastructure	A low-code development platform is part of the corporate infrastructure.	Microsoft Office and the ERP system are part of the corporate infrastructure.
Strategy	Perform business tasks efficiently and effectively with the support of a low-code development platform.	Perform business tasks efficiently and effectively by using carefully developed Excel-based workarounds.

managers who are charged with overseeing low-code/no-code efforts.

1. Exercise Caution When Creating Low-Code/No-Code-Based Workarounds, with or without the Use of a Low-Code Development Platform

The extensive history of system development shows that many projects encounter unanticipated difficulties regardless of the technology that is used and despite the developers' best intentions. Thus, business managers should not view low-code development platforms as a panacea: Though they can help employees create solutions that meet their needs, success is not guaranteed. To achieve their goals, business managers need to engage with frontline employees who will be using the newly deployed (or soon-to-be-deployed) applications, so they can understand the challenges that employees will face and thus identify whether low-code software workarounds can ensure that work can be accomplished successfully. Business managers must approach these engagements with an open mind: They may well learn that a newly (or soon to be) deployed application is not fit for purpose and will likely result in unexpected disasters.

Based on their interactions with frontline employees, business managers should "take stock" of all the issues and problems associated with the new system (and indeed other longstanding issues). Even when a corporate decision has been taken to deploy the new system, it is not too late to take remedial action—for example, by creating low-code/no-code-based workarounds to ensure continued corporate success and customer satisfaction. Business managers should encourage employees to participate in the creation of workarounds by leveraging low-code/no-code techniques. To minimize risk to the organization, employees should carefully assess the operational and security aspects of proposed workarounds.

2. Recognize That a Low-Code Development Platform Is Not Essential for Creating Workarounds

Many of the advantages of workarounds can be attained by using non-platform-based tools, especially Excel. In fact, employees acting as citizen developers may create better solutions if they are not restricted by the rules and other limitations embedded in low-code development platforms. Business managers should therefore not assume that such platforms are necessary to create workarounds. Instead, they should allow suitably qualified employees to consider a variety of different approaches, including formal low-code/no-code tools. These employees should be free to apply a bricolage approach and adopt the tools they deem most appropriate for their context. However, if needed, business managers should ensure that professional IT personnel can act as mentors or quality controllers.

3. Ensure That Workarounds Support Both Local and Corporate-Level Efficiency and Effectiveness

Many of the business rules in the CoreRidge case are built into the warehouse processes and interactions between different groups in the warehouse. Any exceptions, conflicts and other problems were addressed through interpersonal interactions. There is no reason to believe that business rules built into low-code/no-code-based applications would be more effective. An overarching implication is that neither low-code development platforms nor Excel are magic bullets that automatically solve problems in real-world settings.

Solving operational problems in a reasonably sustainable way (i.e., not via one-time workarounds for one-time problems) requires analysis, cooperation and a good fit between the business requirements and the technical solution. Solving operational problems also requires careful oversight. Whether the organization decides to permit low-code/no-code-based solutions or chooses not to deploy a low-code development platform, it is important that the CIO and other C-suite executives ensure that, when workarounds are institutionalized, they provide corporate-wide business benefits and do not cause problems in other parts of the organization.

The implication is that application development efforts should consider whether workarounds will be needed because a proposed application imposes unnecessary constraints, is difficult to use, and/or does not reflect the concerns and interests of the people who actually perform work on a day-to-day basis. There is no

general rule for assigning that responsibility to any C-suite officer in particular because enterprises operate differently. The key issue is that institutionalized workarounds should be monitored to make sure that they support both local and corporate-level efficiency and effectiveness.

4. Use a Portfolio Approach to Low-Code/No-Code Development

The portfolio of possible tools or approaches that can be used for low-code development might include not just a low-code development platform, but also business software tools like Microsoft Excel or even social media applications. Business managers should ensure that the meta-knowledge of how these tools have been applied historically, with success and failure indicators, examples and lessons learned, are shared throughout the organization. Nontechnical aspects, such as the role of brainstorming, focus groups and manager-employee exchanges, should also be included in the portfolio to ensure that low-code development is directed toward solving problems or leveraging opportunities. The portfolio should also include documentation of how combinations of tools and approaches create value.

It is important to note that an overly technical solution is likely to create more problems than it solves. As indicated in Figure 1, though technology lies at the heart of low-code/no-code development and workarounds, it must be supported by interpersonal communication in which problems are identified and discussed before workarounds are conceptualized and crafted. As workarounds are put in place and institutionalized, business managers should ensure they are subjected to continuous evaluation and feedback, improved where necessary, and documented in standard operating procedures and training manuals. Eventually, workarounds may be retired—for example, when new core systems are deployed or business processes are changed.

5. Recognize That a Low-Code Development Platform May Be a Better Option Where Tight Regulatory Control Is Essential

Extensive communication is always required between employees who need a technology solution and those who design and develop the solution. In organizations where a looser security regime is possible, there may be considerable value in permitting employees to create their own solutions or workarounds by leveraging tools such as Excel. But organizations that must adhere to tight regulatory controls will likely need to deploy a low-code development platform to ensure that appropriate operating protocols, governance standards and cybersecurity measures are adhered to, with oversight by the CIO and chief technology and security officers. Ideally, business managers should ensure that such corporate oversight of workaround development and the resulting solutions is in place irrespective of whether the solutions are crafted via a corporately mandated low-code development platform or a less formal Excel-based arrangement.

Concluding Comments

Low-code/no-code development has shown a great deal of promise in many areas. The CoreRidge case vividly illustrates that employees responsible for overseeing processes that might use low-code/no-code capabilities should consider the advantages of both low-code/no-code platforms and non-platform arrangements. Our research implies that corporate systems may be inadequate for work processes, no matter how well-intentioned or carefully designed. Global organizations, in particular, with their multitude of operating locations, local cultural variations, environments and contexts, may never be able to create entirely homogeneous applications that fit all their varied needs. As a consequence, they will likely need to permit some degree of variation, whether through customization or low-code/no-code-based workarounds. Business managers should respond sensitively to the need for these variations and recognize the value of low-code/no-code development, whether supported formally via a low-code development platform or by other less formal workaround arrangements.

In conclusion, we believe that researchers should explore the different dimensions of

low-code/no-code development, including the institutionalization of low-code/no-code-based workarounds. They should neither assume that low-code/no-code work is always platform-based nor assume that a platform is always the best option. A more critical view of the advantages and disadvantages of different forms of low-code/no-code work is needed.

Appendix: Research Methodology and Interview Protocol

We interviewed 31 employees in CoreRidge's Hong Kong warehouse, asking about their use of the company's ERP system. These employees worked in multiple positions and at different levels, ranging from trainee to senior management, in the administration, customer delivery, shipping, logistics, warehouse management, quality control and inventory control functions. Each interview lasted on average 45 minutes, was conducted in Cantonese, and was recorded and later transcribed. The first two authors independently read all of the interview transcripts multiple times.

We applied a thematic analytical technique to the qualitative data obtained from the interviews, following the principles of grounded theory.[37,38] By engaging in a rigorous process of disciplined imagination, we identified major thematic categories in which we documented how workarounds were developed and maintained and how they persisted over time. The low-code/no-code nature of these workarounds became apparent as we explored the use of Excel in workaround crafting and the evolution of the workarounds themselves.

About the Authors

Robert M. Davison

Robert Davison (isrobert@cityu.edu.hk) is a professor of information systems at City University of Hong Kong. His research focuses on the use and misuse of information systems, especially with respect to problem solving and knowledge management in Chinese organizations. He is particularly known for his scholarship in the domain of action research. Robert is the editor-in-chief of *Information Systems Journal* and *Electronic Journal of Information Systems in Developing Countries*. As a researcher and as an editor, he seeks to promote both an inclusive and an indigenous perspective to research.

Louie H. M. Wong

Louie Wong (Louie_Wong@gsm.nucba.ac.jp) is a professor and an associate dean at NUCB Business School, Nagoya University of Commerce and Business, Japan. He specializes in information systems, with a particular focus on the intersection of technology and human behavior. His current research interests include action research, AI in business, blockchain application, digital leadership, digital transformation, social media and supply chain management. With decades of industry experience working for renowned global technology companies before entering the academic world, he integrates practical knowledge with scholarly expertise. Louie was awarded his Ph.D. in information systems by City University of Hong Kong.

Steven Alter

Steven Alter (alter@usfca.edu) is a professor emeritus at the University of San Francisco. Previously, he was vice president of a software startup and has authored four editions of a major information systems textbook. From this experience, his research focuses on developing systems analysis and design methods that business professionals can use on their own and thus help them collaborate more effectively with IT professionals, consultants and vendors. Most of his publications concern the work system method (WSM), work system theory (WST), service systems, and extensions of WST related to workarounds, system interactions, facets of work, cyber/human systems and AI-usage contexts.

37 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.

38 Glaser, B. G. and Strauss, A. *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine Transaction, 1967.